FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

BACHELOR THESIS

Barbora Štěpánková

Generation of Czech Lyrics to Cover Songs

Institute of Formal and Applied Linguistics

Supervisor of the bachelor thesis:	Mgr. Rudolf Rosa, Ph.D.
Study programme:	Computer Science
Study branch:	Artificial Intelligence Bc.

Prague 2024

I would like to thank my supervisor Rudolf Rosa for his valuable guidance and support throughout the project. I am grateful for his help and advice whenever needed, right up to the last moments. I would also like to thank my family, friends and all respondents for their patience and participation in completing the questionnaire for the manual evaluation of our approaches.

metacentrum i guess

Title: Generation of Czech Lyrics to Cover Songs

Author: Barbora Štěpánková

Institute: Institute of Formal and Applied Linguistics

Supervisor: Mgr. Rudolf Rosa, Ph.D., Institute of Formal and Applied Linguistics

Abstract: This thesis explores the topic of generating Czech lyrics to English cover songs. Songs are often adapted to different languages to make them more available to people who do not necessarily speak the language of the original song. During the translation process, however, it is essential to preserve the singability of the text in relation to the melody of the original song, as well as the meaning of the song, so that the translated text fits the context of the original. Currently, such translations are done by hand. We analyze and present the first approaches to solve this problem for Czech through automatic generation using NLP methods. In our work, we create and provide a dataset consisting of pairs of English song lyrics and their official Czech translations. We also provide a dataset of pure Czech song lyrics. We compare the quality of several generative language models. To thoroughly evaluate and analyze their quality, we introduce several automatic metrics and take into account the results of manual evaluation. We find that smaller trained models perform better than larger untrained models. In addition, context is important for the generation of good covers. Finally, we show that our task can be approached from both the translation and generation point of view.

Keywords: natural language processing, text generation, literary NLP

Contents

Introduction **3**

1 Background **4**
1.1 Song lyrics . . . 4
1.1.1 Song sections . . . 4
1.1.2 Cover versions . . . 5
1.1.3 Singable covers and translations . . . 6
1.2 N-gram Language Model . . . 8
1.3 Neural Language Models . . . 8
1.3.1 Transformer block . . . 9
1.3.2 BERT . . . 10
1.3.3 Generative large language models . . . 10
1.3.4 Training and fine-tuning . . . 10
1.3.5 Few-shot prompting . . . 11
1.3.6 Sampling methods . . . 11

2 Related Work **12**
2.1 Poetry and Lyrics Generation . . . 12
2.2 Existing Datasets . . . 12

3 Metrics **14**
3.1 External Metrics . . . 14
3.1.1 BLEU . . . 14
3.1.2 chrF . . . 15
3.2 Reimplemented Metrics . . . 15
3.2.1 Syllable count distance . . . 15
3.2.2 Semantic similarity . . . 16
3.2.3 Phoneme distinct-2 . . . 16
3.3 Our metrics . . . 17
3.3.1 Rhyme Scheme Agreement . . . 17
3.3.2 Rhyme scheme accuracy . . . 18
3.3.3 Syllable accuracy . . . 18
3.3.4 Phoneme repetition difference . . . 18
3.3.5 Keyword similarity . . . 19
3.3.6 Line-by-line keyword Similarity . . . 19

4 Datasets **20**
4.1 Bilingual Song Lyrics . . . 20
4.1.1 Alignment . . . 20
4.1.2 Postprocessing . . . 21
4.2 The Large Czech Songbook . . . 21
4.2.1 Preprocessing . . . 21
4.2.2 Segmentation . . . 22

5 Methodology **24**
5.1 Structure Extraction . . . 24
5.1.1 Syllable Counter . . . 25
5.1.2 Rhyme Scheme Detection . . . 26
5.1.3 Keyword Extraction . . . 28
5.2 N-gram-based framework . . . 29
5.3 Generative neural language models . . . 30
5.3.1 Fine-tuning models to generate whole lyrics . . . 30
5.3.2 Fine-tuning models to generate lyrics line-by-line . . . 32
5.3.3 Generating whole lyrics with Few-Shot prompting . . . 34
5.4 Postprocessing . . . 34
5.4.1 Choose Best . . . 34
5.4.2 Remove and add stopwords . . . 36
5.5 Evaluator . . . 37

6 Experiments and Evaluation **38**
6.1 Random baseline and target values . . . 38
6.1.1 Machine translator baseline . . . 42
6.2 Models . . . 43
6.3 Automatic evaluation . . . 44
6.3.1 Raw outputs . . . 47
6.3.2 After postprocessing . . . 51
6.4 Manual Evaluation . . . 58
6.4.1 The method . . . 58
6.4.2 The questionnaire . . . 59
6.4.3 Results . . . 59
6.5 Discussion . . . 65

7 Implementation and usage **66**
7.1 Command line interface . . . 66
7.1.1 Implementation . . . 66
7.1.2 Usage . . . 66
7.2 Web application . . . 68
7.2.1 Implementation . . . 68
7.2.2 Usage . . . 68

Conclusion **71**

Bibliography **73**

List of Figures **77**

List of Tables **79**

Introduction

Cover songs are created all around the world. Some of the covers stay very close to the original, while others may tweak the lyrics a bit. Some covers translate the lyrics into another language. There are two very prominent reasons why someone would want to translate a song into their language: either they like the song and want to bring it closer to the speakers of their language, or the song is part of a musical film or a musical theatre show that is being adapted. Nowadays, both of the cases are handled by a human translator or a lyricist. When creating the cover in a different language, many formal requirements must be met: the lyrics must fit the melody, for musicals the lyrics must fit the context in meaning and for musical films specifically the singer must be able to lip-sync the song with the original motion picture. The process of writing these lyrics is ineffective and in the age of natural language processing models, automatic methods should be explored to see how these models handle the task.

In this thesis, we address the problem of generating lyrics to Czech covers of English songs. The task of generating lyrics is quite underresearched and we are the first to tackle it for Czech or similar languages. To the best of our knowledge, no adequate dataset for this task has been published. Additionally, it is not clear how to measure the quality of the covers. Our work aims to analyze the issue, highlight the challenges, and propose initial approaches to the automatic generation of Czech covers.

We collect and process the necessary data and conduct initial experiments. We propose some possible metrics and analyze their behaviour on various kinds of data. We try several approaches to Czech cover generation and subsequently evaluate these approaches to determine which avenue of research to take next. In this work, we compare various generative models with different training strategies.

In Chapter 1 (Background), we will introduce the necessary terminology and knowledge for understanding the rest of the work. In Chapter 2 (Related work), we will summarise the research already done on this topic. In chapter 3 (Metrics) we will introduce the metrics we use in this work, both our own and the ones proposed by various papers. In chapter 4 (Datasets), we describe the process of making, as well as the final form of two datasets. In chapter 5 (Methodology), we describe in detail the methods used in each step of the proposed pipeline. In chapter 6 (Experiments), we run many experiments and thoroughly evaluate them both by the automatic metrics and the human evaluators. In chapter 7 we describe the high-level overview of the pipeline as well as the overview of the web application attached to this work, and document the usage.

1. Background

In this chapter, we will define terms and provide the necessary background for the reader to understand the rest of this work. First, we will explain all necessary terminology associated with songs and their lyrics. Then, we will briefly introduce language models and methods of leveraging the model outputs.

1.1 Song lyrics

Song lyrics are words that are sung to a song. Usually, the lyrics map onto the melody of the song and flow with the changes of the rhythm and the intonation. Even though we see the importance of melody concerning lyrics, due to the lack of proper data, we will explicitly work only with the lyrics.

By deciding to work purely with the lyrics of the songs, our task in some ways similar to poetry translation and generation. In some cases, similar approaches and tools can be used for either poetry or lyrics generation/translation, however, in other cases, the tasks are substantially different and have to be approached differently, especially concerning the formal constraints of poems and song lyrics.

1.1.1 Song sections

In this work, we decided to focus on individual song sections rather than whole songs. A music section can be defined as a complete, but not independent, musical idea. [Bye, 1993] By this definition, song sections are commonly intro, verse, pre-chorus, chorus, bridge, outro etc. For our task, only the sections with lyrics are of interest to us. These usually are verse and chorus, occasionally bridge or pre-chorus. Each of these sections holds a different role: while the chorus is usually repetitive and simple to remember, the role of a verse or a pre-chorus is to tell a story or lead up to the chorus of the song.

We use the terms *section* or *lyrics section* to address the lyrics of any coherent music section of a song, regardless of whether it is a chorus, a verse or any other song section. In the following example, we can see one verse and one chorus of the same song. We will handle each of these sections independently.

The snow glows white on the mountain tonight
Not a footprint to be seen
A kingdom of isolation
And it looks like I'm the queen

Let it go
Let it go
Can't hold it back anymore
Let it go
Let it go
Turn away and slam the door

1.1.2 Cover versions

'A "cover" is typically defined as a recording of a song that was first recorded by someone else.' writes Magnus [2022] in his book 'A Philosophy of Cover Songs'. A similar definition is provided by Cusic [2005]: 'The definition of a "cover" song is one that has been recorded before.' However, this topic is much more complex. Magnus introduces 5 problems of cover songs. They deal with the definition of an original version based on the time of the first recording or a performance, as well as on the authorship of the song. For us, the most relevant problem is the fifth problem about whether when the lyrics are changed, the new song still qualifies as a cover. According to Magnus [2022], it does.

In Czechoslovakia in the 1970s, many well-known foreign songs were covered and in the process translated into Czech or Slovak. These covers sometimes stayed close to the original meaning of the text, but frequently concentrated more on phonetic similarity and singability. One of the famous examples may be "Mám Styl Čendy" *[I have Čenda's style]* by Karel Gott, covering Elton John's "I'm Still Standing". Later on, these covers continued, but mostly for satyric reasons (for example parodies by the band Těžkej Pokondr). A comprehensive list of Czech Cover versions can be found on Wikipedia[1], to date counting 5118 cover songs.

A subset of Czech cover songs is considered to be song translations, oftentimes made for a musical theatre or a musical film. These, except for a few exceptions, do not appear on the Czech Cover Versions List (which contains mainly parodies), but due to having the same melody, according to Magnus [2022], these songs also qualify as covers. The challenge in song translation is mainly in conveying the meaning without sacrificing the essence of the song. As stated by [Low, 2003] *"In song translation, the constraints are imposed by the pre-existing music"*.

Based on that, [Franzon, 2008] introduces the following *Five choices in song translation*:

1. *Leaving the song untranslated*
2. *Translating the lyrics but not taking the music into account*
3. *Writing new lyrics to the original music with no overt relation to the original lyrics*
4. *Translating the lyrics and adapting the music accordingly – sometimes to the extent that a brand new composition is deemed necessary*
5. *Adapting the translation to the original music*

Each of these choices is adequate for a different task. For making different language cover versions, choices 3, 4 and 5 are relevant, as they satisfy both the 'different language' part and the 'Cover' part. When making subtitles, option 2 is perfect. When making a musical film adaptation, option 1 might be a good choice. In this work, we decided not to take music directly into account, so it is out of the question to adapt the music to the lyrics (choice 4), which leaves us with choices 3 (results in parodies) and 5 (results in singable translations).

[1]Wikipedia, The Free Encyclopedia, Seznam Českých coververzí zahraničních skladeb. [accessed 18-March-2024]

As Franzon [2008] suggests, song translation is not a translation in the true sense of the word, it is more of making a cover version while taking the original meaning, mood and context into account. In this work, we will use the terms *song translation* and *song cover* interchangeably, always meaning that lyrics were rewritten from the original language to a target language.

1.1.3 Singable covers and translations

In this section, we will introduce the main principles that make a song translation singable, almost entirely adapted from Peter Low. In his paper, Low [2003] introduces the *pentathlon principle* of singable translations, which consists of five criteria: *singability*, *sense*, *naturalness*, *rhythm* and *rhyme*. Low describes the *pentathlon principle* as an attempt to balance the above-mentioned criteria, the same as athletes competing in pentathlon must compete in five events, and optimize their scoring overall. Even though Low writes purely about translations, as stated in Section 1.1.2, the line between non-parodic cover versions and song translations is very thin and therefore the principle applied to both cases.

Rhythm and Singability

Rhythm and singability are closely connected and even though Low describes them separately we decided to refer to them together as *singability*. *Singability* is a property of a text, meaning that the text can be easily and comfortably mapped onto a certain melody. To do that, the text has to have a certain rhythm, mainly dictated by syllable counts and subsequently by syllable stress.

Keeping the same syllable counts per line as the original lyrics is highly desirable for easily mapping lyrics onto the melody, but it is not strictly necessary, as the melody can be slightly tweaked to accommodate the lyrics. The standard in Western music is to put one syllable per each note, however, in some cases, it is possible to sing one syllable over several notes instead. It is called melisma, and a famous example in English is the final *" Hallelu-u-u-u-jah"* of the chorus in the song *Hallelujah* by Leonard Cohen, or in Czech the *"voda hu-u-čí po luči-i-nách"* from the Czech national anthem. The opposite example may be splitting one long note into several shorter ones to fit more syllables in. Another option for adapting lyrics to the melody may be slurring syllables together, and not pronouncing them properly. One of the common examples of this phenomenon may be the schwa sound, called the reduced vowel, which is very popular in English. The schwa sound is, for example, the *'a'* in *'about'* or the second *'e'* in *'everything'*. Simply put, the speaker can decide to pronounce it fully, but oftentimes it becomes so reduced that the syllable created by this vowel can be skipped over. [Flemming, 2009]

Low [2003] also emphasizes the importance of not only keeping the syllable counts, but also choosing the correct words in the correct parts of the lyrics based on the stress of the word: *"Common English words like 'it' and 'the' can easily be sung to a short note, preferably a quaver, but can scarcely be held for a minim. It is true that singers sometimes pronounce 'it' as 'eeet' while still making sense; but often this will not match the music."* The stressed syllables usually fall on down beats and on notes held for a longer time, but detecting these patterns in song lyrics might prove to be difficult, as *"Rhythm in songs is not the same as a* *metre in traditional poetic scansion."* [Low, 2003] It is much more song-specific, dependent on the dynamics of the melody as well as the stress patterns of the lyrics.

Sense

Unless making a parody, it is desired that the cover version has some semantic similarity with the original version. However, the constraints of the song justify and sometimes even require a modification of meaning.

Naturalness

The lyrics of the cover should sound natural. It is okay to switch up word order a bit in favour of other criteria, for example to enforce rhyming or singability, but according to Low [2003]: *"A song text must communicate effectively on first encounter. ... A singable translation is not worth making unless it is understood while the song is sung."* When making a cover, it is better to omit some meaning of the song to express the few core thoughts naturally.

Rhyme

Rhyme oftentimes makes up the structure of the song, pairing up the lines and sections and tying it all together. On the other hand, keeping up the rhyme scheme perfectly may come at the cost of ignoring the sense or the naturalness of the lyrics, which is not ideal. Low [2003] mentions that: *"In this case the rhymes will not have to be as perfect or numerous as in the original, and the rhyme scheme need not be observed strictly."*

Compared to poetry, songs do not require to have perfect rhymes (identical phonemes at the end of the lines). Slant rhymes (similar but not identically sounding), identical rhymes (the same word in both sound and sense) and even words that can be mispronounced to sound similarly are considered rhyming in song lyrics. Another difference is that while poetry is heavily based on rhyme schemes considering rhymes at the end of the line, a lot of rhymes in song lyrics are hidden by repetition of a sequence of phonemes within a line or a section (internal rhyme).

perfect rhyme	*Patience unmoved! no marvel though she* ***pause*** *They can be meek that have no other* ***cause***
slant rhyme	*Hey yo, I'm just like my* ***country*** *I'm young, scrappy and* ***hungry***
identical rhyme	*Cause way down deep inside we've got a* ***dream*** *I've got a* ***dream***
internal rhyme	*Of* ***cour****se it's hard to have inter****cour****se* *over* ***four sets*** *of cor****sets***

Table 1.1: Types of rhymes commonly appearing in song lyrics

1.2 N-gram Language Model

The n-gram model is a probabilistic model that predicts the likelihood of a word given the previous $n-1$ words in a text sequence.

It operates on the principle of Markov assumption, which states, in the context of n-grams, that the probability of a word w_k depends only on the previous $n-1$ words instead of the whole history of the words preceding w_k. Let us consider a simple bi-gram model with $n = 2$. With $w_1, w_2, \dots, w_{k-1}$ being a word sequence, to predict the probability of w_k following this sequence, normally the probability would be counted as $P(w_k|w_1 w_2 \dots w_{k-1})$. The Markov assumption for $n = 2$ says that:

$$P(w_k|w_1 w_2 \dots w_{k-1}) \approx P(w_k|w_{k-1}) \tag{1.1}$$

To construct an n-gram model, we obtain the probabilities by calculating the maximum likelihood estimation (MLE). The probability of a word w_k using n-grams of size n given the previous words $w_{k-n+1} \dots w_{k-1}$ is computed as:

$$P(w_k|w_{k-n+1} \dots w_{k-1}) = \frac{Count(w_{k-n+1} \dots w_{k-1} w_k)}{Count(w_{k-n+1} \dots w_{k-1})} \tag{1.2}$$

The n-gram model is constructed by tokenizing a corpus of text into words and then calculating the probability of each n-gram appearing in the corpus of text according to Eq. 1.2. To combat the problem of words appearing in a previously unseen context, smoothing is used. The simplest way to do smoothing is to use the Laplace smoothing. In Laplace smoothing, 1 is added to each word of the vocabulary (each word appearing in the corpus of text the model is trained on) before normalizing the counts into probabilities. The new probabilities are calculated as follows, with V signifying the vocabulary size:

$$P_{Laplace}(w_k|w_{k-n+1} \dots w_{k-1}) = \frac{Count(w_{k-n+1} \dots w_{k-1} w_k) + 1}{Count(w_{k-n+1} \dots w_{k-1}) + V} \tag{1.3}$$

1.3 Neural Language Models

In this section, we are going to assume that the reader has basic knowledge about neural networks. We will introduce the transformer architecture and mention self-attention and its importance. Then we will discuss the training and fine-tuning process of transformer-based models, as well as inference, sampling and prompting methods.

Most modern neural language models are based on the transformer architecture that is based on the encoder-decoder architecture. When used for causal language modelling (predicting the next token following a sequence of tokens), the transformer gets a sequence of words as the input and returns a prediction of the following token. The encoder part of the transformer focuses on understanding and extracting information from the input and creating a contextualized representation of it, while the decoder takes this representation and is responsible for generating new tokens.

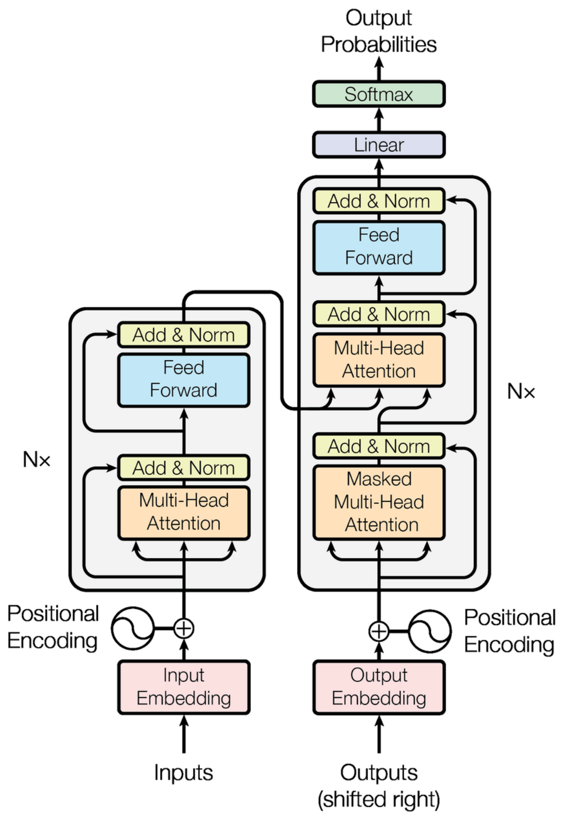


Figure 1.1: The original transformer model architecture from the paper *Attention is all you need* by Vaswani et al. [2017]

Nowadays, the majority of relevant language models are based on transformers. Transformer-based models are split into three categories: encoder-only architectures, encoder-decoder architectures and decoder-only architectures. All have in common that they consist of stacks of transformer blocks.

1.3.1 Transformer block

A transformer block maps a sequence of input vectors $(x_1, \ldots, x_n)$ to a sequence of output vectors $(z_1, \ldots, z_n)$. The block consists of a multi-headed self-attention layer and a layer of small feed-forward neural networks, one for each token, with layer normalization after each operation and residual connections around each operation (see Fig. 1.1).

The layer normalization normalizes the outputs of both self-attention and the feed-forward neural network to keep the values in a reasonable range to avoid exploding and vanishing gradients. Residual connections allow passing of information from lower layers to the higher layers more efficiently.

Self-attention mechanism looks at the surrounding token representations and integrates the information from them into the current token representation, to better build the meaning of the token based on the surrounding context. It helps the model with learning the relations between words and parts of the input, as well as focusing on specific parts of the input while generating the following token.

The attention mechanism calculates a weight for each element of the input, indicating the importance of that element for the current prediction. These weights *are then used to calculate a weighted sum of the input, which is used to generate the prediction. Multihead self-attention is a specific type of attention mechanism where the model pays attention to different parts of the input sequence in order to make a prediction. It means the model is looking at the input sequence multiple times, and each time it is looking at it, it is focusing on different parts of it.* [Jurafsky and Martin, 2024]

Attention can be either masked or unmasked, depending on whether the model can look into the future or not. In encoder transformer blocks, the attention should look into the future, in the decoder transformer blocks, upon inference the model only knows the preceding sequence of tokens, therefore during training, masking all tokens occurring after the current one is necessary. Then, the model is performing a masked self-attention.

1.3.2 BERT

Bidirectional Encoder Representations from Transformers (BERT) [Devlin et al., 2018] is an example of a model implementing the encoder-only architecture. BERT is able to learn complex connections from input texts, and create concise word and sentence embeddings.

1.3.3 Generative large language models

Generative large language models typically use the decoder-only architecture, with multiple decoder-type transformer blocks being stacked onto each other. Around this sequence of transformer blocks, a necessary interface must be built.

First, the input of the model has to be tokenized by a pre-trained tokenizer. The tokenizer segments the text into subwords from a vocabulary of a fixed size. There are many different tokenization methods available. One of the common options is using the BPE (Byte-Pair Encoding) algorithm [Sennrich et al., 2015], which gradually merges the most frequent character sequences until we achieve the desired vocabulary size. The tokenizers can encode text into tokens as well as decode the output tokens back to text.

The tokenization is followed by an embedding layer. The token embeddings are assigned to individual tokens, as learned during the training phase. We also provide information about the positions of the tokens, both absolute and relative to surrounding tokens, to the self-attention algorithm, which is done by creating a position embedding.

Finally, the tokenized and embedded input is passed through the sequence of the transformer blocks. The probabilistic distribution over the tokens that could be generated next is computed from the output of the transformer blocks by the language modeling head composed of a linear layer outputting the logits and a softmax layer outputting probabilities of individual tokens from the vocabulary.

1.3.4 Training and fine-tuning

Both training and fine-tuning of generative transformer-based language models use a self-training algorithm, which in each time step t tries to predict the next word. *We train the model to minimize the error in predicting the true next word* *in the training sequence, using cross-entropy as the loss function.* [Jurafsky and Martin, 2024]

$$L_{CE} = -\sum_{w \in V} P_t(w) log \hat{P}_t(w) \tag{1.4}$$

Cross-entropy loss measures the difference between a predicted probability distribution $\hat{P}_t(w)$ and the correct distribution $P_t(w)$.

1.3.5 Few-shot prompting

Few-shot prompting is a technique of teaching models to generate a specific answer to a specific question via prompting. It is done by showing several question-answer pairs as an example in the prompt before posing the question we want the model to respond to, as can be seen in Table 1.2[2].

Prompt:	This is awesome! // Positive This is bad! // Negative Wow, that movie was rad! // Positive What a horrible show! //
Model output:	Negative

Table 1.2: Example of 3-shot prompting a model to output a sentiment of a sentence in one word

1.3.6 Sampling methods

Lastly, let us briefly mention different sampling methods. Sampling is the process of choosing a token to generate based on the probability distribution provided by the model. While the simplest decoding method is greedy sampling (choosing the most probable token), it does not often provide satisfactory results.

Top-k sampling means choosing the most probable k tokens, normalizing their probabilities to sum up to 1 and choosing out of these tokens according to their probability. Similarly, top-p sampling chooses the top most probable options, but instead of fixing a number, it chooses the most probable p percentage. [Holtzman et al., 2019] In general, the higher the p, the more diverse the generated vocabulary.

In temperature sampling, the probabilistic distribution is reshaped before sampling, by dividing the logits u by the temperature parameter t, computing the probabilities as:

$$P = \text{softmax}\left(\frac{u}{t}\right) \tag{1.5}$$

Low temperature values usually mean more logical text, while high temperature lets the model choose more freely.

[2]Example of few-shot prompting from https://www.promptingguide.ai/techniques/fewshot

2. Related Work

2.1 Poetry and Lyrics Generation

Poetry and lyrics generation tasks are closely connected and while each has its specifics, many of the main ideas are similar.

There are two main approaches to poetry and lyric generation. Generation according to a given structure, or generation via a constrained translation of a source section. One could say these two options are equivalent to when a human lyricist is writing lyrics for a song with only melody and no previous lyrics (or without understanding the previous lyrics) and trying to make a singable translation of the song.

The main difference between poetry and lyric generation is that while poetry is heavily based on formal constraints like rhyme schemes and metric patterns, lyric generation is heavily based on a pre-existing melody.

Many different approaches have been taken to poetry and lyrics generation. After some rule-based attempts that have been proposed as early as in the 1970s [Farringdon, 1970], (for example [Díaz-Agudo et al., 2002] with a rule-based approach to generating Spanish poetry), Manurung [2004] proposed the use of evolutionary algorithms.

Genzel et al. [2010] constrain a statistical MT system to return translations of a specific length, meter and rhyme scheme. This is done as a tradeoff to the overall translation quality. Malmi et al. [2016] propose a deep learning approach to rap lyrics generation and introduce a rhyme-density metric. Hopkins and Kiela [2017] use recurrent neural networks with rhythmic constraints for poetry generation. RNNs are also used by Watanabe et al. [2018], who proposed a melody-conditioned language model based on mapping word boundaries onto breaks in the melody.

Lee et al. [2019] takes a new approach to lyrics-to-melody generation by splitting the task into two subtasks: lyrics-to-rhythm (duration of the notes) and rhythm-to-melody (pitches of the notes), using an LSTM seq2seq model for each. A lyric-melody-aligned dataset was used to train both of these models.

Transformer-based auto-regressive language models were used to generate song lyrics with a predefined format, with special symbols to improve the models' performance [Li et al., 2020, Zou et al., 2021, Lo et al., 2022]

Song translation for tonal languages (in this case Chinese) is addressed by Guo et al. [2022], adapting the tone of the lyrics to the changes in melody. Special tokens specifying the desired number of syllables and a rhyme scheme, as well as word boundary tokens were added by Ou et al. [2023].

Recently, a new metric for evaluating the singability of translated lyrics was proposed by Kim et al. [2023b]. The metric does not rely on the melody and considers just a pair of song lyrics and their linguistic properties.

2.2 Existing Datasets

To the best of our knowledge, there are no datasets containing Czech and English aligned song lyrics, Czech lyrics with melodies available, or even English lyrics aligned with melodies, to which a Czech translation could be found.

In general, datasets of song lyrics are rare. Goto et al. [2002] made a database of song lyrics with aligned melodies, however, the majority of the song lyrics are in Japanese. Benito-Santos et al. [2023] put together a dataset of Spanish song lyrics without melodies for comparing semantic similarity across songs. A Chinese hand-aligned dataset of lyrics and melodies extracted from midi was created by Lee et al. [2019]. Guo et al. [2022] uses monolingual and unaligned data in English and Chinese as a train set, and only a few aligned English-Chinese song lyrics with melodies as a test set. An English-Korean singable lyric translation dataset aligning Korean and English lyrics line-by-line and section-by-section was proposed by Kim et al. [2023a]. Approximately 89% of the dataset consists of K-pop song lyrics.

3. Metrics

Metrics play a pivotal role in our task. They are needed throughout the whole solution, from the analysis of datasets, through training of the language models, to the final evaluation of results.

We experimented with many metrics, but in the end, settled on the ones described in this section. We divide them into three categories based on the origin of the metric. The first category contains external metrics, that are well known and commonly used to evaluate results. The second category contains reimplemented metrics, that we reimplemented and adapted based on descriptions in previous works by other authors. The last category contains our metrics which we invented based on the need to measure that specific information and the lack of an adequate metric for that specific task.

As stated in Section 1.1.2, a defining quality of a cover version is that the main musical idea (such as melody) stays mostly unchanged, even when the lyrics change. Covers do not have to be translations, but unless a satyric cover is desired, it is usually considered to be better when covers have at least a similar topic or are conveying the same feeling as the original. Our goal is to propose metrics to measure foremost the singability of the covers, rhyme of the covers and semantic similarity between the cover and the original, copying the modified *pentathlon principle* presented in Section 1.1.3. Even though song translations and cover versions differ from traditional translations of unconstrained text, tracking the scores of translation metrics tells us about the naturalness and correctness of word order.

3.1 External Metrics

3.1.1 BLEU

BLEU (Bilingual Evaluation Understudy) introduced by [Papineni et al., 2002] is a metric commonly used to evaluate the quality of machine translations against one or more (usually human) reference translations. It was developed to address the challenge of automatically measuring the quality of translations without human judgment.

BLEU is based on the concept of overlapping n-gram precision. For each n, the ratio of the number of n-grams in the candidate translation that also appear in any reference translation, to the total number of n-grams in the candidate translation is calculated. From these precision scores, the weighted average is computed. Usually, n is from 1 to 4, with the weights for the weighted average being (0.25, 0.25, 0.25, 0.25). There is also a brevity penalty, penalizing shorter translations, that could artificially increase the precision score.

Due to BLEU being a weighted average of precision multiplied by a brevity penalty which is between 0 and 1, the BLEU score will always give a number between 0 and 1. A higher BLEU score (closer to 1) indicates that the machine translation is closer to the human-translated references in terms of n-gram matches.

3.1.2 chrF

The Character n-gram F-score (chrF) metric introduced by [Popović, 2015] is another method used to evaluate the quality of machine translations compared to human reference translations. chrF is based on the idea of comparing character-level n-gram overlap between the candidate translation and reference translations, instead of word sequences as in BLEU (see 3.1.1). This ensures that chrF works well when evaluating translations in languages with complex morphology.

The chrF metric computes the F-score, which combines precision and recall of character n-grams. Precision measures the proportion of character n-grams in the candidate translation that also appear in the reference translation, whereas recall measures the proportion of character n-grams in the reference translation that are found in the candidate translation. The F-score is calculated as the harmonic mean of precision and recall:

$$F_\beta\text{-}score = \frac{(1+\beta^2)\cdot precision \cdot recall}{(\beta^2 \cdot precision) + recall} \tag{3.1}$$

chrF gives results between 0 and 1, closer to 1 means that the candidate translation has a higher overlap with the reference translation and, therefore should be better.

3.2 Reimplemented Metrics

The topic of computational song lyrics analysis is quite underresearched and there are not many resources on metrics comparing the singability of songs. Four metrics are proposed in “A Computational Evaluation Framework for Singable Lyric Translation” paper by Kim et al. [2023b], each covering a different aspect of creating singable song translations, specifically for English, Japanese and Korean. One focuses on syllable counts, one on the meaning of the lyrics and two on the high level overview of the song, comparing individual sections of both songs in the correlation of repetitiveness and similarity of the sections within the song, effectively checking whether choruses map onto choruses and verses on verses. As we focus on generating individual sections without the knowledge of the rest of the song, we reimplement only the two metrics that use the individual song sections independently of the rest of the song.

3.2.1 Syllable count distance

Maintaining similar syllable counts between the original and the new lyrics for each line is essential for the two sets of lyrics to fit the same melody. Syllable count distance measures the closeness of lengths of each two lines.

Let us denote a pair of lyrics consisting of n lines as $X = \{x_1, \ldots, x_n\}$ and $\tilde{X} = \{\tilde{x}_1, \ldots, \tilde{x}_n\}$, where each element represents one line. Let *syl* be a syllable counter function. For example, if the first line of the English lyrics X is $x_1 =$ *"Do you want to build a snowman"* and the corresponding line in Czech lyrics would be $\tilde{X}$ is $\tilde{x}_1 =$ *"Nechceš postavit sněhuláka"*, then $syl(x_1) = 8$ and $syl(\tilde{x}_1) = 9$.

[Kim et al., 2023b] define the line syllable count distance between a pair of lyrics X and $\tilde{X}$ as follows:

$$SylDist(X, \tilde{X}) = \frac{1}{2n} \sum_{i=1}^{n} \left(\frac{|syl(x_i) - syl(\tilde{x_i})|}{syl(x_i)} + \frac{|syl(x_i) - syl(\tilde{x_i})|}{syl(\tilde{x_i})} \right) \tag{3.2}$$

This metric returns a float, that could be interpreted as the percentual difference in length of the two texts. Therefore, it can take values from zero to infinity, usually staying in the range of 0 to 1, as it is unusual that one of the compared lyric lines is more than 100% longer than the other one.

3.2.2 Semantic similarity

To show how much the meaning of an original song deviates from the meaning of the cover song, Kim et al. [2023b] propose the *semantic similarity* metric. First, the contextual embeddings of each lyrics are obtained using a pre-trained Sentence BERT model[1] and then the cosine similarity between the embeddings is calculated. As this model was trained for English, the lyrics in the different languages have to be machine-translated before obtaining the embeddings. Kim et al. [2023b] explore the semantic similarity of the two songs based on the size of the subsections that are being compared. They concluded that due to the various word orders of different languages, and the varying number of syllables needed to express the same thoughts in different languages, the best option is to compare the semantic similarity section-wise, not line-wise. After obtaining the similarity for each section of the song, it is averaged by a weighted average, where the weights are the number of lines in each section over the number of lines in the whole song.

We adapted this metric for just one section, as we generate only one section at a time. We are also translating the Czech sections into English using Lindat translator [Popel et al., 2020] before obtaining the embedding of the whole section. From the section embeddings, cosine similarity is calculated. This cosine similarity is the output of the metric for each section separately.

3.2.3 Phoneme distinct-2

Phoneme distinct-2 is a monolingual metric, that describes the diversity and repetitiveness of a section of a text, as the ratio of the number of distinct phoneme bigrams to the total number of phoneme bigrams[2][3]. Originally, it was introduced by [Li et al., 2015] where it measured diversity in terms of word bigrams, later the metric was adapted for phonemes by [Kim et al., 2023b]. Even though this metric is monolingual, Kim et al. [2023b] use it in the two metrics comparing the section-wise structure of the whole song. We are using it in one metric of our own (3.3.4), so we thought it appropriate to describe it properly here. The phoneme distinct-2 (*dist2*) of section X_i is computed as:

$$dist2(X_i) = \frac{\#distinct\ bigrams\ in\ X_i}{\#bigrams\ in\ X_i} \tag{3.3}$$

[1]sentence BERT https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2
[2]Czech phonetic transcription https://github.com/lukyjanek/phonetic-transcription
[3]English phonetic transcription https://pypi.org/project/eng-to-ipa/

Phoneme distinct-2 can take values between 0 and 1. Lower values indicate higher repetition and vice versa.

3.3 Our metrics

The reimplemented metrics from Kim et al. [2023b] focus on the linguistic relationship between English, Japanese and Korean lyrics. Also, from the *pentathlon principle* of singability (see Section 1.1.3), they tackle only the criterium of *sense* and one aspect of *singability.* In this section, we will propose two metrics measuring *rhyme* (*Rhyme scheme agreement* and *Rhyme scheme accuracy*), two metrics measuring *singability* (*Syllable count accuracy* and *Phoneme repetition difference*) and two metrics measuring *sense* (*Keyword similarity* and *Line-by-line keyword similarity*).

None of the metrics we propose measures the *naturalness* of the lyrics, as we find that it can only be measured by human evaluators. Evaluation of naturalness will have to be done manually. We tried implementing a metric comparing the *singability* in a sense of the rhythm of the Czech and English lyrics by using tools made for detecting metre in poetry. However, as stated in Section 1.1.3, song lyrics usually do not follow any specific metre, as the stress falls on the syllables accentuated by the melody, not necessarily where the stress should correctly be. Furthermore, stress in Czech and English acts differently, is defined differently and overall is not well transferable from one language to the other.

For the whole of the following section, we will refer to a pair of lyrics consisting of n lines as $X = \{x_1, \ldots, x_n\}$ and $\tilde{X} = \{\tilde{x}_1, \ldots, \tilde{x}_n\}$, where each element represents one line.

3.3.1 Rhyme Scheme Agreement

Rhymes are very prominent in both Czech and English texts, compared to Eastern poetry and songs, therefore, we decided to include a rhyme scheme agreement as a metric. We assume that one of the two schemes being compared is the ideal rhyme scheme we are trying to achieve, while the other is our attempt to get as close as possible to the desired rhyme scheme. Let's call them *desired* and *new* rhyme schemes. This metric is not symmetrical.

Our metric consists of two components put together by weighted average. The first component is a cross-lingual, checking how many specific rhymes were kept in the new scheme during the translation (recall), and the second component is checking only whether enough rhymes were kept, not dependent on their location. This comes from [Low, 2003], where is stated that having fewer rhymes, or rhymes in different places is better than having no rhymes.

Let us denote the desired rhyme scheme as a graph R with the indices of lines $\{x_1, \ldots, x_n\}$ as nodes, and the new rhyme scheme as a graph $\tilde{R}$, with the indices of lines $\{\tilde{x}_1, \ldots, \tilde{x}_n\}$ as nodes. An edge between the nodes i and j in the rhyme scheme R means that the lines x_i and x_j rhyme. $Edg(R)$ returns the set of indices of the rhyming lines in the rhyme scheme R. Both the desired and the new rhyme scheme have to have the same amount of nodes, meaning the same amount of lines in the song section.

The first part is computed as a recall, and the second part is computed as the minimum of rhymes per section out of the two compared sections, divided by the number of rhymes in the desired rhyme scheme. Let $\#Edg(R)$ be the number of edges (rhymes) in the rhyme scheme.

$$RhymeAgree(R,\tilde{R}) = \alpha \cdot \frac{Edg(R) \cap Edg(\tilde{R})}{Edg(R)} + (1-\alpha) \cdot \frac{min(\#Edg(R), \#Edg(\tilde{R}))}{Edg(R)} \tag{3.4}$$

The best score of the metric is 1 when both of its components achieve a score of 1. The first part is equal to 1 when all of the edges from the desired rhyme scheme are also in the new rhyme scheme. Extra edges in the new rhyme scheme don't influence the score in any way. The second part of the score is 1 when there are at least as many edges in the new rhyme scheme as in the desired rhyme scheme, regardless of the position. This way, when the new rhyme scheme has 0 overlapping edges with the desired scheme, but the same amount of edges as the desired scheme, it can still obtain a score of $(1-\alpha)$ as it is better than having no rhymes at all.

3.3.2 Rhyme scheme accuracy

Similarly as in Section 3.3.1, let us denote the rhyme schemes of two compared song sections as graphs R and $\tilde{R}$, where the nodes of the graph are the indices of lines $\{x_1, \ldots, x_n\}$ and $\{\tilde{x}_1, \ldots, \tilde{x}_n\}$. An edge is between two nodes $idx(x_i)$ and $idx(x_j)$ when the lines x_i and x_j rhyme. $Edges(R)$ returns the set of indices of the rhyming lines in the rhyme scheme R. The compared sections have to have the same number of lines, so the rhyme schemes can be mapped onto each other by the line indices. Rhyme scheme accuracy is computed as:

$$RhymeAcc(R,\tilde{R}) = \frac{Edges(R) \cap Edges(\tilde{R})}{Edges(R) \cup Edges(\tilde{R})} \tag{3.5}$$

Effectively computing the number of common edges divided by the number of all edges

3.3.3 Syllable accuracy

Syllable accuracy is the accuracy of syllable counts of the English verse being equal to the syllable counts of the Czech verse computed over all n lines. Let *syll* be a syllable counter function. Then, syllable accuracy is calculated as follows:

$$SyllAcc(X,\tilde{X}) = \frac{\sum_i^n 1 \text{ if } syl(x_i) == syl(\tilde{x}_i) \text{ else } 0}{n} \tag{3.6}$$

3.3.4 Phoneme repetition difference

Phoneme repetition creates a sense of rhythm and trying to minimize the difference between the phoneme repetition of the two compared sections means getting closer to the desired rhythm and repetition. Phoneme repetition is similar to the rhyme density measure used for measuring the quality of rap lyrics [Malmi et al., 2016]. Second, phoneme repetition difference combats the nature of generative language models to repeat themselves, by getting very low scores on the very repetitive output sections, therefore making the difference higher.

Phoneme repetition difference is the difference of *phoneme distinct-2* values of the two song sections (see Section 3.2.3), computed as:

$$\mathit{PhonDiff}(X, \tilde{X}) = |\mathit{dist}2(X) - \mathit{dist}2(\tilde{X})| \tag{3.7}$$

The disadvantage is that because the phoneme distinct-2 is computed as the number of distinct bigrams over the number of all bigrams, we do not have any information about the distribution of the repeating bigrams in the section. This means that a very rhythmic and natural section could have the same phoneme distinct-2 as a section where one half consists of the same phoneme bigram repeating many times and the other half does not have any repetitions at all. However, these are special cases that are not common.

3.3.5 Keyword similarity

Keywords are a simple way of summing up the meaning of text. We are using the *keyword similarity* metric to compare the similarity of the keywords of the English input section and the Czech output section.

The same as in *semantic similarity* in section 3.2.2, the Czech section is machine-translated to English before the keyword extraction, the keywords are joined by commas and then *semantic similarity* of these joined keywords is computed.

3.3.6 Line-by-line keyword Similarity

Even though Kim et al. [2023b] showed that line-wise similarity is lower than section-wise similarity, when generating each line separately, the line-wise similarity is needed.

We also implement the line-wise similarity by comparing the keywords for each line the same as *keyword similarity* in Section 3.3.5, with the only exception of extracting the keywords, joining them and getting the *semantic similarity* for each line separately. The final score is obtained by averaging the scores of all the lines of the section.

4. Datasets

For our task of Czech lyrics generation for cover songs, several different datasets are needed. As already discussed in Section 1.1.3, melody and lyrics are inseparably tied together. It would be ideal to obtain a dataset containing aligned lyrics in Czech, lyrics in English and their melody. Unfortunately, as discussed in Section 2.2, such a dataset does not exist, and to the best of our knowledge, there are no available data to make one as such.

In this section, we describe two datasets we created to the best of our abilities to help us with the task of Czech cover lyrics generation. One dataset consists of Czech and English aligned bilingual song lyrics and the other one is monolingual, containing a large number of Czech song lyrics.

4.1 Bilingual Song Lyrics

In this section, we will present our *Bilingual Song Lyrics* dataset, containing 649 aligned song sections in Czech and English. We will show the process of the dataset creation, as well as the dataset analysis.

Most of the songs were originally written in English and translated from English to Czech by professional human translators. A subsection of the lyrics was originally written in French and our dataset contains the English and Czech adaptations of the French original (also done by human professionals). All of the song lyrics were taken out from songs performed in popular musical theatre shows or musical films, therefore we assume that the quality of the translations is high, with a focus not only on *singability*, *naturalness* and *rhyme*, but also with a heavy focus on *sense*. All of the song lyrics gathered were freely downloadable from the internet.

The dataset contains 649 verses from 69 songs from 10 musicals, namely: Encanto, Frozen, Frozen 2, Grease, The Jungle Book, Les Miserables, The Lion King, The Little Mermaid, Moana and Tangled. All of these 69 songs were obtained in both Czech and English, aligned to map onto each other line by line, having as similar as possible number of syllables on each line, split into sections and then saved into JSON format.

4.1.1 Alignment

There are several problems concerning the alignment of two texts, which supposedly map perfectly onto each other. The first problem could be in the assumption of the existence of the perfect mapping. Sometimes human translators sacrifice the exact matching of syllables for meaning. Also, the lyrics we obtained sometimes (but not always) included lines of unsung dialogue and just the fact that the dialogue is unsung poses another problem because, with the melody absent, the syllable constraint is gone. The dialogue is often unmappable, but not distinguishable based on the text only, from the rest of the lyrics. Other than dialogue, in some songs there are character names in front of sections, when the song is sung by multiple characters. Another problem with the syllable counts per line could originate from inaccurate analysis of the text by the syllable counting function.

Assuming the existence of a perfect mapping, there is still the problem of the translators taking liberty with the line breaks and rhyme schemes. Take for example the Czech and English versions of the chorus of Les Miserables' "Do you hear the people sing?" (see Fig. 4.1) where it can be seen that although the syllables of the verse match perfectly, the lines are unalignable unless the second half of the verse is merged into one line.

Do you hear the people sing
Singing the song of angry men
It is the music of the people
Who will not be slaves again,

Slyš tu píseň zástupů
ze slzavého údolí
chcem tady žít
už máme bídy dost
i běd a svévolí

Figure 4.1: English and Czech version of "Do you hear the people sing?" from Les Miserables with unalignable lines

We experimented with many lyric-aligning methods, including a method of alignment using dynamic programming, where the goal was to find the minimum loss mapping, but due to the above-mentioned problems, all of these methods proved unsuccessful. In the end, the lyrics were aligned by a naive semi-automatic aligning function looking for the smallest difference in syllables between two lines, looking at the current line, split line, or current and following line combined, taking the best combination. In case the difference was higher than 1 syllable per pair of lines, the program stopped and waited for a manual input. In that case, the lines in question were either correctly aligned or in the case of the dialogue or the name tag, removed. This system proved to be much more efficient than relying purely on an algorithm or aligning the lyrics fully by hand.

4.1.2 Postprocessing

After aligning the lyrics, the text was lower-cased, the commas, full stops and line breaks removed from the ends of the lines, and the individual verses were saved into a JSON format.

4.2 The Large Czech Songbook

The Large Czech Songbook is a corpus of song lyrics extracted from the *Velký zpěvník* webpage[1]. It contains 17599 mainly Czech songs from 1381 interprets, both recent and from the previous century. The final dataset contains 77478 individual song sections with corresponding analysis, saved in JSON format.

4.2.1 Preprocessing

In this section, we will describe the process of cleaning the data. We got the scraped data from the Velkyzpevnik webpage from Martin Popel in a raw text file, containing not only the lyrics but also the metadata of the songs, chords,

[1]https://www.velkyzpevnik.cz/

suitable instruments that can play the song, and information about the intro and outro melodies.

Our first step in cleaning the data was lowercasing and gathering the most common words in the raw text file and deciding on suspiciously frequent words, like ‘předehra’, ‘CD’, ‘text’ or ‘sólo’. With high probability, these words denote the song’s metadata, which is not of importance to us and our task. We removed these words, or the whole lines containing these words, depending on the nature of the word itself. The full list of words that triggered line deletion can be found on GitHub[2]. Another type of symbol that appeared frequently was ‘...’, often preceded by just a few words from the refrain, signalling the refrain repetition. These words together with the dots were also removed. The last type of trigger words were chords. These were taken care of by a series of regex, catching the most common writings of chords. In case a chord was written in a rare form, most often it was caught by one of the line-deleting words.

As previously stated, only most of the songs from the Large Czech Songbook are actually in Czech. To filter away lines written in a different language than Czech, we use langdetect.[3] If neither of the resultant languages is Czech, the line is deleted. This also worked as a ”nonsense lines” remover. If the text written on the line could not be classified as Czech, we did not want it in the corpus.

Finally, all special symbols and diacritics except ’,’ and ’.’ were deleted.

4.2.2 Segmentation

Next, we will describe the process of segmenting the raw data into sections of appropriate size.

Some of the song sections were separated from the rest of the text by two line breaks, but the rest of the sections were often separated just by an indent in the text or by the number of the verse and a colon. The preprocessed text contained 24 blocks of text longer than 100 lines without any double line breaks, with two of them being well over 400 lines. Furthermore, sometimes a song section had two line breaks after every line, making it seem that each line of the song section was its own song section.

Based on these observations, determining the lower and upper limits of the length of the song sections is necessary. We choose the lower boundary as 3 lines per song section and the upper boundary as 8 lines per song section. Next, we implement a recursive function for splitting a song section into two, as well as a recursive function for joining two song sections together.

The criterion of the split is the *Phoneme distinct-2* metric. For each possible splitting point, we calculate the phoneme distinct-2 for each of the two newly created sections. The split with the lowest sum of the phoneme distinct-2 scores is chosen and the sections proceed to the next recursion step. The effect of this is that lines with similar phoneme patterns will be in the same section. Similar phoneme patterns create a sense of rhythm and compactness, which is highly desired in a song section.

Not only the length of a section but also the line length was varied. The longest line was 1218 characters long, the median line length was 31 characters

[2]https://github.com/stepankovab/GenerationOfCzechLyricsToCoverSongs
[3]https://pypi.org/project/langdetect

per line. Where possible, the long lines were split on punctuation. Long lines without punctuation were deleted.

5. Methodology

Our task is to generate Czech lyrics based on the provided English lyrics. The generated Czech lyrics should be singable to the same melody as the English input without prior knowledge of the melody, as well as have similar semantic meaning.

This task can be principally approached from two different directions, both carrying with them their advantages and disadvantages: the machine translation approach and the language modelling approach. We carefully considered both of these directions. As explained in section 1.1.2, song lyrics translation is not a typical translation task. Each of the proposed choices in song translation [Franzon, 2008] requires a different generation method. Generating a new text independent of the original meaning comes to a purely form-constrained generative language modelling, translating the lyrics directly and adapting the melody comes to using a plain machine translation and finally, adapting the translation to fit the melody can be done by constrained machine translation. Dataset analysis of the bilingual data (see Section 6.1) shows that according to standard machine translation metrics, the human-translated lyrics are quite distant from machine-translated lyrics (BLEU = 0.03 and chrF = 0.16). This shows that lyric translation seems to be rather far from the standard setting for machine translation, despite the lyric pairs having decent semantic similarity scores. As language models can have constraints not only concerning the structure of the lyric but also the meaning of the lyrics, we decided to choose the path of language models, rather than machine translation.

The high-level overview of our solution to the task is as follows. First, we need to analyze the input texts and extract all the important structural and semantic information. Next, we generate the output text based on the analysis of the input text. We take several approaches to both generation and postprocessing, all are described in the following subsections. Finally, we evaluate the quality of the output based on the input by automatic evaluation metrics.

We chose to experiment with different language models and different input information to find out which combination yields the best results. We implemented the naive N-gram model with a heuristic generation function to fill in the given lyric structure. We are also using multiple neural language models with several different training approaches. All of the approaches we experimented with require a specific input that encodes the structure of the original lyrics. The model input differs based on the model-specific training but the information needed stays the same.

5.1 Structure Extraction

Structure extraction is a crucial step that takes place multiple times in the pipeline. First, we need to extract the structure of the English input to try and guess the important aspects that make the English text singable. Then, we need to extract specific aspects of the structure for the postprocessing step. Finally, we use structure extraction in the evaluation of the final generated text, extracting structure from both the Czech and English text. In our work, we understand ‘structure’ in a broad sense, encompassing both formal (melody) and semantic (meaning) characteristics of the lyrics.

In this section, we will describe the methods of extracting individual characteristics from both Czech and English lyrics. Originally, the idea was to extract melody from the lyrics, however, all mentions of successful lyrics-to-melody generation worked with a dataset of aligned lyrics-melody pairs, which we did not manage to obtain. Nevertheless, several characteristics of lyrics closely tied to either the melody or the meaning need to be extracted: the number of syllables for each line, the rhyme scheme, keywords summarising the whole lyrics and keywords summarising each line of the lyrics.

5.1.1 Syllable Counter

One of the most important aspects of song lyrics is the matching of syllables to the melody. To ensure our generated outputs follow the rhythm of the original input as closely as possible, it is crucial to have an adequate syllable counter for both the English input and the Czech output.

Czech syllable counter

We used a rule-based algorithm for the Czech syllable division proposed by [Štindlová, 1968] as an algorithm to decide on a word division in automatic typesetting. We heavily based our implementation of the syllable counter on a previous implementation of this paper, called ‘Sekáček’, made as a year project at our university. [Macháček, 2014]

The syllabification process of ‘Sekáček’ can be divided into two parts. First, a word mask is created for each word to be syllabified. The mask consists of three types of symbols: a vowel symbol, a consonant symbol and a special inseparable symbol, that denotes a character that could be wrongly separated from the rest of the inseparable characters by the algorithm (e.g. ‘o’ in ‘ou’ or ‘h’ in ‘ch’). The Syllable-forming instances of consonants ‘r’ and ‘l’ are masked as vowels. Second, the mask is split according to a few simple splitting rules. After mapping the mask back onto the word, we get the desired syllabification of the word.

We adapted the mask-creating and mask-splitting functions from ‘Sekáček’ and added a few rewriting rules to handle exceptions. These rules are applied right before the mask creation. For example, rewrite ‘osm’ to ‘osu’ or ‘sedm’ to ‘sedu’, artificially adding a vowel to indicate a syllable (only in cases where the ‘m’ is not followed by a vowel: ‘sedmdesát’ gets rewritten but ‘sedmikráska’ stays the same). Exceptions concerning unstressed prepositions ‘k’, ‘v’, ‘s’ and ‘z’ are solved by prepending these prepositions to the following word if it exists and syllabifying the joined words as one.

English syllable counter

For English, we tried multiple syllable counters from various Python libraries for natural language processing. None of them gave the syllabification we required, mostly due to expecting a fully pronounced schwa sound in syllables, therefore giving us more syllables than are pronounced in the lyrics. More on this topic is discussed in section 1.1.3. It proved impossible to change just the intended pronunciation of the syllable-forming schwa sound, and after some trial and error, we decided to extend the Czech syllable counter to English.

We realised that there is no use for the syllabified output and that we are interested in the syllable count only. Also, because Czech is mostly pronounced the same way as it is written, we can take the phonetic transcription of the English text[1], add a few rewriting rules and parse it the same way we would parse a Czech written text. Concerning the schwa problem, because we don't have the melodies available, we decided to assume that when a syllable transcribed to IPA reads [vər], as in 'forever' or 'every', it is rewritten to read [vr]. Because of the syllable-forming 'r' in Czech, when [vər] is at the end of the word, the syllable counter catches it as a syllable and when it is in the middle of a word, it skips it just as most native speakers would.

By running a script for comparing syllable lengths of pairs of lines from the human-translated dataset and stopping when the lengths did not match, we filtered out the majority of mistakes, leaving only mismatched lines when the lyric counts differed. We changed the phonetic transcription to be some kind of 'Czech' phonetic transcription. For example, rewriting 'ay' to 'aj', or 'oy' to 'oj'. This avoids the problem with 'y' being pronounced as [ɪ] in Czech, or changing 'ks' to 'x' to avoid getting a word split in the wrong place. Some errors may slip through, but considering any of the syllables can be uttered quickly, or stretched across multiple notes, we concluded that it is an acceptable risk.

5.1.2 Rhyme Scheme Detection

The rhyme scheme detector lets us see the rhyme schemes of both Czech and English lyrics so we can extract the structure at first and then check how well the structure was filled.

In this section, we describe three different rhyme detectors: an external one (RhymeTagger), our own (RhymeFinder) and an external one with a framework built around it (Same-word RhymeTagger). We show that creating rule-based and database-based rhymers or using rhymers made for poetry is not ideal, as rhyming in songs is freer than rhyming in poetry (see section 1.1.3). The best option is to use an external rhyme detector that achieves good scores on poetry and build a rule-based framework around it to handle the rhymes frequently used in song lyrics specifically.

RhymeTagger

RhymeTagger [Plecháč, 2018] is a collocation-driven method of discovering rhyme schemes in poetry. It includes pre-trained models for both Czech and English. While RhymeTagger successfully recognises a majority of rhymes, it fails to detect for example identical rhymes, as they are not usually accepted as rhymes in poetry, or slant rhymes that are very typical for song lyrics.

Same-Word RhymeTagger

Same-word RhymeTagger uses RhymeTagger as a base. RhymeTagger obtains a rhyme scheme and then the rhyme scheme is modified to consider the same words

[1]https://pypi.org/project/eng-to-ipa/

as rhyming. This is done by unifying the tags of each two lines that end with the same word.

RhymeFinder

RhymeFinder is our rhyme detector that implements different methods for both Czech and English texts. We developed it in the hope that it will be able to better discover rhyme schemes in song lyrics, as lyrics have more relaxed rhyming requirements than poetry (see section 1.1.3). The method of extracting the rhyme schemes differs based on the language of the text. For Czech, we take a simple rule-based approach due to Czech being written mostly in the same way it is being read, while for English we use a web database of rhyming words from a web page specialising in helping people with songwriting.

Czech RhymeFinder

The Czech part of RhymeFinder extracts a *rhyme key* from each line, specifically from the last syllable of each line, by masking the last syllable according to the masking rules. These rhyme keys are compared and based on their matching, a rhyme scheme is made.

While this approach is definitely consistent and easily catches words with the same ending, some rhymes do not get marked, due to rhyming on a more phonetic base, even though the written version is a bit different. Consider the following song section (Figure 5.1) from *Surface Pressure* by Lin-Manuel Miranda, where it feels like the rhyme scheme of the Czech version should be AAABB, but the extracted rhyme scheme is ABBCD. The words *peří* and *věřím* are not registered by our rhyme detector, due to the extra *m* at the end of one of the rhymes.

Or simple pleasure	1.	*Mít chvíli peří*
Instead we measure	2.	*Já tomu věřím*
this growing pressure	3.	*A stejně měřím*
Keeps growing keep going	4.	*jak stoupá tíseň hloupá*
'Cause all we know is	5.	*co mě zkouší spoutat*

Figure 5.1: section from *Surface Pressure* from the *Encanto* Musical, written by Lin-Manuel Miranda and its Czech translation

The mask is created as follows: first, all 'ch' phonemes are replaced by a single letter 'h'. If the syllable is longer than three letters, it is shortened to contain just the last three letters. If it is shorter than three letters, it is padded from the left, so the syllable is at the end of the rhyme key. Similarly sounding consonants (e.g. 's' and 'z', or 'g' and 'k') and vowels with the same pronunciation, or pronunciation differing only by the length of the sound (e.g. 'a' and 'á', or 'y' and 'i') are rewritten as the same symbol. Lastly, if the first position of the rhyme key is occupied by a vowel, the vowel is deleted and replaced by padding. The same applies when there is a consonant followed by a vowel at the beginning of the rhyme key. In that case, the consonant is replaced by the padding. (see Table 5.1)

last word	rhyme key
peří	*řI
věřím	*Im
měřím	*Im
hloupá	*BA
spoutat	*AD

Table 5.1: Mapping of endings to a rhyme key for Czech RhymeFinder

English RhymeFinder

Due to the inconsistent spelling of English phonemes, it would be quite difficult to make a rule-based rhyme scheme detector for English. Therefore we used an online dictionary of rhyming words provided by a web API[2]. Each line is represented only by the last word of the line. For each of these words, we get a list of potential rhymes. If any of the line endings appear in a list obtained from another line ending, those two lines rhyme. The main disadvantage is that due to checking for whole words, the rhyme detector can not detect made-up words or less common words that rhyme but are not in the database of potential rhymes.

Comparison of rhyme detectors

We compared the rhyme detectors' performances on the Bilingual Song Lyrics dataset 4.1. While our RhymeFinder performs competitively to or better than the standard RhymeTagger, we found that this is mostly due to RhymeTagger failing to detect identical rhymes. We therefore devised the Same-Word RhymeTagger setup, which seems to perform best in our setting.

5.1.3 Keyword Extraction

Keywords are the easiest way to summarise a text and give a general idea of what the text is about. In this section, we are going to discuss the used techniques for keyword extraction.

For English, we used KeyBERT [Khan et al., 2022], which uses BERT embeddings and searches for words in a provided text, that are the most similar to the text itself according to cosine similarity. The most similar words are the keywords that characterize the text the most.

For Czech, as we did not manage to find a suitable keyword extractor, we translated the Czech lyrics into English using the Lindat translator [Popel et al., 2020] and extracted the keywords the same way as we did from an English text, using KeyBERT. Then, we translated these keywords back into Czech. Even though this many machine translations might seem unnecessary, an English machine translation of both the Czech lyrics and Czech keywords is needed for measuring their semantic similarity, so it would have to be made either way. In a real use case of keyword extraction, we are keeping the keywords of the Czech lyrics both in Czech and English.

For some types of downstream tasks, it is necessary to have keywords for the whole section, and for others, it is better to have keywords for each line

[2]Rhymebrain by Steve Hanov https://rhymebrain.com/api.html [Accessed 21-March-2024]

separately. The methods described above can be used for both cases, depending only on whether the input text is the whole section or a single line.

5.2 N-gram-based framework

In this section, we will describe the framework used to generate text following a certain structure using an N-gram language model as a base. This framework is purely monolingual, expecting already extracted structure and generating a Czech output according to that structure.

The N-gram model is trained on 1.25 million word tokens from fiction books, and 1.82 million word tokens of song lyrics from the Large Czech Songbook dataset (see Section 4.2). The model is good at predicting the following word, and due to being trained on a decent amount of song lyrics, sounds quite poetic. However, all of the structural constraints have to be manually reinforced during decoding.

The framework generates the final lyrics line by line, but the N-gram model is always prompted by the previous line to maintain continuity. During the decoding, the N-gram-based generator takes the desired number of syllables for each line and the desired rhyme scheme. Then, for each line, until enough syllables are generated, a recursive function keeps choosing a word based on the probability distribution provided by the N-gram model. When the line reaches the correct number of syllables and (if specified) the correct ending syllable, the line gets added to the already generated output. When the line contains more syllables or a wrong ending, words are removed from the line, as well as from the distribution of possibilities provided by the N-gram model and a new word is chosen according to the new modified probability distribution. As long as there is a solution satisfying the structure in the probabilistic space of the N-gram model, that solution will get outputted. If there is no such possibility, the last word of the previous line gets repeated and the n-gram model space is searched again. If even that fails (for example for a lack of a rhyming word) a line without a rhyme, or with an imperfect syllable count will get outputted instead.

8 A	*někdy se pro mě není ten*	*a to byla ona a to*
8 B	*ten který se na mne snese*	*co se stalo v minulosti*
8 A	*jsem stár a byl jsem oblíben*	*minulosti to jsem já kdo*
8 B	*že je to ale na mne se*	*kdo z nás má na to já jsem ti*

Figure 5.2: Two examples of lyrics generated by the N-gram-based model with a given structure of 8 syllables per line and rhyme scheme of ABAB

The biggest advantage of this framework is that it is straightforward, simple and predictable. The generated lyrics keep the structure well. However, there are disadvantages to this approach. The lyrics are not very singable, as the framework often chooses monosyllabic words without stress for the last word of the line, due to the model itself having no idea about the position the generated word will take. It is also difficult to inject any meaning into the lyrics.

We concluded that the n-gram-based framework is not suitable for our task of making singable covers while trying to preserve meaning, mainly because all of the decisions of the framework are based on simple rules and N-gram probabilities and the framework lacks the depth required for this task. We keep the n-gram model as a demo for the web application (see Section 7.2), but we will not explore it further in this work.

5.3 Generative neural language models

In this section, we will discuss various methods used for leveraging the output of neural language models to be closer to our desired results. We will look at the different approaches we took during training and inference.

We tried three different approaches, two using fine-tuning techniques and one using a few-shot technique. We experimented with various prompts and provided the models with different information before generating the output. The impact of these changes on the quality of generation can be found in section 6.

To fine-tune a language model to generate a specific output, we need to provide the model with enough examples of the text we want to imitate. If we want the generated text to keep a structure based on our requirements, it is also necessary to point out these structural elements in each training example.

Coming back to the pentathlon principle of singability, a singable text has to have rhyme, sense, singability (including rhythm) and naturalness (see Section 1.1.3). Rhyme and sense are taken care of by the encoded extracted structure, and singability is from a big part defined by syllable counts, which are also extracted from the English input structure. Only naturalness is not specifically defined. We discovered that the criterium of naturalness was fulfilled by fine-tuning the model on poetic texts of song lyrics, which by default sound as natural as we want the generated texts to sound. As we do not have any metric checking the naturalness of the lyrics, it will have to be manually evaluated after generation, to assess how successful we were.

5.3.1 Fine-tuning models to generate whole lyrics

The first method of fine-tuning is to leverage the model to generate the whole lyrics of a song section at once.

Because of the structure extraction step, there is no need for bilingual data for training. During Inference, the English input will serve only as the source of the structure, and the model will be able to generate a Czech version of the lyrics based only on that structure. Therefore, we are using the Large Czech Songbook dataset 4.2 for the training.

From each verse in the Large Czech Songbook, we extract the syllable counts, rhyme scheme and keywords and format the training examples into a prompt containing a subset of this information, followed by the annotated lyrics (see Table 5.2). There are four prompt types depending on the chosen subset of extracted structural elements: syllables only, keywords only, syllables and keywords, and syllables, keywords and a rhyme scheme. This way, we can see the influences of individual added tags, as well as train a model without overwhelming it with too much information. During fine-tuning, the loss is calculated as cross-entropy on the probability of the following word in the training example being generated by the model.

syllables (WS)	
prompt:	5 7 6 7 #
lyrics:	5 # krása a tvůj smích
	7 # mě pod košilí zebou
	6 # trpím jak mladej mnich
	7 # a chtěl bych tě vzít sebou

keywords (WK)	
prompt:	4 # mnich krása smích zima košile #
lyrics:	krása a tvůj smích
	mě pod košilí zebou
	trpím jak mladej mnich
	a chtěl bych tě vzít sebou

syllables and keywords (WSK)	
prompt:	5 7 6 7 # mnich krása smích zima košile #
lyrics:	5 # krása a tvůj smích
	7 # mě pod košilí zebou
	6 # trpím jak mladej mnich
	7 # a chtěl bych tě vzít sebou

syllables, keywords and rhyme scheme (WSKR)	
prompt:	5 7 6 7 # A B A B # mnich krása smích zima košile #
lyrics:	5 # A # krása a tvůj smích
	7 # B # mě pod košilí zebou
	6 # A # trpím jak mladej mnich
	7 # B # a chtěl bych tě vzít sebou

Table 5.2: Training examples for fine-tuning LLM's to generate whole song sections with different information available

During inference, we extract the structure from the English text, make the prompt the same as we do in the training examples and let the model generate the rest. After generation, we isolate the lyrics from the output by removing generated tags.

5.3.2 Fine-tuning models to generate lyrics line-by-line

We encountered problems when trying to generate a whole song section with a specific rhyme scheme, and a perfect solution seemed to be to borrow the technique from the N-gram model (see Section 5.2) and generate each line separately. We noticed that because of the structure tags, the lyrics are not generated in one piece anyway, so the connectivity of the lyrics is already broken, so generating each line separately should not have a big impact on the naturalness of the lyrics, while improving the rhyme scheme.

We trained two models for each prompt type: one for lines that end with a specific syllable and a second for lines that don't have the final syllable specified yet. The final song section is obtained by joining the individual generated lines.

The training process was the same as described in section 5.3.1. The prompt types for a line-by-line generating model were: syllables-only, keywords-only, syllables and keywords, syllables and unrhymed line, and all of the aforementioned prompt types with the added ending tag (rhyming syllable hint) to reinforce rhyme schemes. We will discuss individual prompt types in the rest of the section, see Table 5.3 for examples.

The syllables-only prompt was created by prepending the syllable count of the line to the line with a division tag in between. The syllables-and-ending prompt was created by prepending both syllable count and the last syllable, or the last three letters of the syllable if the syllable was longer than three characters, to the line, also having a division tag between each information. When including keywords and other information in the prompt, we put the keywords on a previous line, just to have even more division between the keywords and the actual desired output.

The unrhymed prompt types attempt to utilize the ability of language models to rewrite text with different words while keeping the meaning. The prompt contains a translated text by a machine translator with the number of syllables the translated text has on one line, and the number of syllables the desired output should have on a new line. Of course, if we want to generate a line with a certain ending, the ending has to be included in the prompt besides the syllable count.

During training when we work with monolingual data, we can obtain text with the same meaning but different wording by translating each line to English and back. The meaning stays but the words used almost always change. We are trying to show in the training data that the most important thing is to rewrite the line to satisfy the syllable count we desire. Because of that, when the double-translated line has the same syllable count as the desired line, we copy the double-translated line to the output, effectively teaching the model to copy the input if the syllable count is the same. When line endings come into the picture, both syllable count and ending have to be matched to use the input line as the desired line. During inference, we translate the input English line to Czech and consider that to be the "unrhymed" line. (see Table 5.3 for examples)

	syllables (LS)
example:	7 # mě pod košilí zebou
	syllables with ending (LS)
example:	7 # bou # mě pod košilí zebou
	keywords (LK)
example:	chladná košile # mě pod košilí zebou
	keywords with ending (LK)
example:	chladná košile # bou # mě pod košilí zebou
	syllables and keywords (LSK)
example:	chladná košile # 7 # mě pod košilí zebou
	syllables and keywords with ending (LSK)
example:	chladná košile # 7 # bou # mě pod košilí zebou
	syllables and unrhymed (translated) (LST)
example:	9 # ať je mi pod košilí zima 7 # mě pod košilí zebou
	syllables and unrhymed (translated) with ending (LST)
example:	9 # ma # ať je mi pod košilí zima 7 # bou # mě pod košilí zebou

Table 5.3: Training examples for fine-tuning LLMs to generate song sections line by line with different information available

Reinforcing rhyme

Lastly, we will talk about choosing when to use the model reinforcing line endings and when not to. The structure extractor extracts the structure out of the English input, including the rhyme scheme. Based on the rhyme scheme, a dictionary is created with the letters denoting the rhyming lines as keys (e.g. for the rhyme scheme of 'ABAB', the keys are 'A' and 'B'). When we want to generate a line which has no value assigned to the corresponding letter in the dictionary, we use the model focusing only on length and/or meaning. After generating the line, the last syllable is added to the dictionary under the corresponding letter of the rhyme scheme. When a line's corresponding key has a value already assigned, the model that is reinforcing endings is used, with the ending from the dictionary in the prompt.

5.3.3 Generating whole lyrics with Few-Shot prompting

For few-shot prompting, we use the same prompt types as in section 5.3.1 when fine-tuning a model to generate whole verses of lyrics. Effectively we build a prompt consisting of a few examples of the training data, ending with the prompt used for the fine-tuned model generating whole verses.

5.4 Postprocessing

Although it is valuable to consider just the raw model output to estimate the quality of a model, we need to polish the output to have a more consistent lyrics generator.

We implemented two kinds of postprocessing functions that can be applied separately or after one another on the same text. The first one runs the generation n times and then chooses the best out of the n outputs. The second takes the generated output and removes or adds stopwords to come as close to the desired syllable count as possible.

5.4.1 Choose Best

The first of the two postprocessing functions gets n generated outputs and only the best one is returned. There are two variants to this function. One is choosing the best out of n song sections, the second is choosing the best out of n lines. The first one is used when the whole section is generated in one call, the second one is used when the section is generated line-by-line.

Choose best line

In this section, we will describe the process of choosing the best line out of n generated lines. When generating lyrics line by line, we decided to always choose the best line before continuing with generating the next line, mainly to avoid branching the dictionary of rhymes by keeping several options for each letter of the rhyme scheme (see Section 5.3.2).

When choosing the best line, we take syllable distance (see 3.2.1), semantic similarity (see 3.2.2) and matching of the ending (if the ending is known) into account. To decide on the importance of these separate aspects, we introduce *tolerance* value for syllable distance and for semantic similarity. A tolerance value states what is an acceptable value of the metric in question. Based on the tolerance value and the metric value, a *penalty* P is computed. Let us denote syllable distance as *SyllDist* and semantic similarity as *SemSim* and the generated line as x. The syllable distance penalty is computed as:

$$P_{SyllDist}(x) = \frac{SyllDist(x)}{SyllDist\ tolerance} \tag{5.1}$$

This means that when the syllable distance is equal to the syllable distance tolerance, the penalty is 1. When the syllable distance is 0, the penalty is 0. The computation is similar to the semantic similarity penalty, with the exception that semantic similarity has the best value at 1 and not at 0.

$$P_{SemSim}(x) = \frac{(1 - SemSim(x))}{(1 - SemSim\ tolerance)} \tag{5.2}$$

The penalty for a wrong ending of the line is either 1 if the line does not rhyme with the desired ending or 0 when it does.

The penalty of each line is a sum of these three penalties. The line with the lowest penalty is deemed the best. We experimented with the values of the tolerances until we found such values that the best line according to the penalties is consistently the same line we would have chosen as the best. These default tolerance values are: $SyllDist\ tolerance = 0.1$ and $SemSim\ tolerance = 0.6$.

Choose best section

To choose the best out of n song sections, we compute syllable distance (see 3.2.1), rhyme scheme agreement (see 3.3.1), semantic similarity (see 3.2.2) and phoneme repetition difference (3.3.4) between the input text and each of the generated outputs. The same as when choosing the best line (see Section 5.4.1), we introduce *tolerance* value and compute the *penalty* P from that value for each of the considered metrics.

The penalty of syllable distance is computed as in Eq. 5.1 and the penalty of semantic similarity is computed as in Eq. 5.2, both with the whole section as an input. We denote the generated section as X and rhyme scheme agreement as *RhymeAgree*, its penalty is computed as:

$$P_{RhymeAgree}(X) = \frac{(1 - RhymeAgree(X))}{(1 - RhymeAgree\ tolerance)} \tag{5.3}$$

Lastly, the penalty of phoneme repetition difference (denoted as *PhonDiff*) is computed as:

$$P_{PhonDiff}(X) = \frac{PhonDiff(X)}{PhonDiff\ tolerance} \tag{5.4}$$

For each generated song section, these four penalties are computed and then summed together. The song section with the lowest sum of the penalties is chosen as the best one. Again, we experimented with the *tolerance* values, until we found a combination that yields results that correlate the most with our choice of the best section. These values are: *SyllDist tolerance* = 0.1, *SemSim tolerance* = 0.6, *RhymeAgree tolerance* = 0.4 and *PhonDiff tolerance* = 0.15

The reason for a quite low rhyme scheme agreement tolerance and a strict phoneme repetition difference tolerance is the proneness of language models to repeat themselves. When the generated section contains four of the same lines, the rhyme scheme agreement will be perfect, but unless the original English input was also this repetitive, the phoneme repetition difference will be high, and some other option with imperfect rhyme scheme, that is overall better, will be preferred.

5.4.2 Remove and add stopwords

In this section, we will describe the second postprocessing function that corrects the length of the generated outputs. The function gets one generated song section and the desired line lengths on the input and tries to correct the length of the line as much as possible by removing or adding stopwords.

Remove Stopwords

For the removal of stopwords, we are using a Czech stopwords list found on a web page dedicated to text processing tools[3], from which we removed stopwords that when removed change the sentence too much in the grammatical sense. If the length of the output line is two or more times than the desired length, the stopwords-removal algorithm is not even run, because removing half of the syllables by removing stopwords would result in nonsense, if even possible. There is also the option of keeping the last word intact if a rhyme scheme depends on that word.

The algorithm itself is based on identifying stopwords within the output and their subsequent recursive removal while searching for the minimal difference between the desired length and the current length of the modified output. If the algorithm is faced with the choice of removing more or longer stopwords and returning the output n syllables shorter than the desired length, or removing fewer stopwords and returning the output n syllables longer than the desired length, it always chooses to remove more and return shorter than desired output. This is because in that case, the output will be passed to the stopwords-adding section of this postprocessing function.

Add Stopwords

For adding stopwords, we take a simple rule-based approach. If the difference in syllables is 1, we randomly choose a line prefix from 5 common, meaningless words: *a, tak, že, co, dál.* If the difference in syllables is 2, we choose a line prefix from: *že prý, a tak, a pak, copak, no tak.* When the difference is 3 or more syllables, we fill in this difference by a prefix consisting of *"na"* copied as many times as needed (but at least three times), resulting in lines starting with *"na na na na"*, singing without lyrics, as is common in Czech songs.

[3]Czech stopwords list https://countwordsfree.com/stopwords/czech

5.5 Evaluator

In this section, we are going to introduce an evaluating function, crucial to determining how good the lyric generation is. It is also used for automatically evaluating and comparing different generation methods.

The evaluator gets the English input text and the Czech output text as the input and returns the results of ten metrics computed on the Czech and English inputs. The requirements the input must meet are that the lengths of the Czech and English inputs are the same in terms of lines. Because our models concentrate on generating only one verse at a time, it is assumed that just one verse of a song is passed to the evaluator.

The metrics considered are syllable distance 3.2.1, syllable accuracy 3.3.3, rhyme scheme agreement 3.3.1, rhyme scheme accuracy 3.3.2, semantic similarity 3.2.2, keywords similarity 3.3, line-by-line keywords similarity 3.3, phoneme repetition difference 3.3.4, BLEU2 score considering word bigrams 3.1.1 and chrF score 3.1.2, all described in respective sections.

Translation scores modification

All metrics except BLEU and chrF compare the two song sections directly. Both BLEU and chrF are popular scores used for evaluating translation by comparing the translated text with reference translations. When dealing with song lyrics, there are two options of what we could consider to be the reference translation: it could be the human-translated singable adaptation of the song or a pure machine translation of the original lyrics.

We decided to use the machine translation of the English lyrics as the reference and the Czech generated lyrics as the candidate solution. This way, we can evaluate the translation metrics even on texts that do not have a human-translated cover version already. The biggest disadvantage is that the translation scores measure the translation quality in both sense and word order, and the word order of song lyrics is oftentimes modified to fit the melody. However, if we considered the human-translated adaptation to be the reference translation, there would be no direct link between the reference and candidate, as the information about the desired meaning of the song section comes from the English original, not from the already adapted Czech Cover.

Concerning the BLEU score, word 4-grams are considered by default. Due to the 4-gram BLEU having a score of zero even on the bilingual human-translated dataset, we lowered the n to 2 and we are using BLEU2. That way the BLEU score gives more relevant results. We also tried looking into a metric more adequate for evaluating high-inflexion languages like Czech: the Character F-score.

6. Experiments and Evaluation

In this chapter, we are going to present the results of our experiments, both in terms of the metrics, as well as in examples of lyrics. First, we will introduce our baselines and targets. Then, we will evaluate the results of the experiments with automatic metrics. Lastly, we are going to discuss the results of the manual evaluation of our experiments outputs.

6.1 Random baseline and target values

Both the random baseline and the target values are measured on the Bilingual Song Lyrics dataset (see section 4.1).

To obtain the target values, we evaluate the English-Czech pairs of song sections from the Bilingual Song Lyrics dataset by the evaluation function (see section 5.5). These target values are not the highest values that the individual metrics of the evaluation function can achieve, they are values achieved by human lyricists and translators on songs translated for real-life projects.

To measure the random baseline, we first need random pairs of lyrics. These are obtained by shuffling the English song sections of the Bilingual Song Lyrics dataset and pairing them back up with the unshuffled Czech song sections. The longer of the song sections from the pair gets truncated so both sections have the same number of lines, and then are evaluated by the evaluation function. Even though the choice of the pairs is random, the compared song sections are still from the same domain of musical song lyrics. We considered comparing plain text with song sections as the baseline, but this idea proved to be inadequate because most of the metrics are specifically made for song lyrics.

The average values of the individual components of the evaluation function can be seen in the following table:

	worst	**Baseline**	**Target**	**best**
Syllable Distance	inf	0.65	0.03	0
Syllable Accuracy	0	0.10	0.83	1
Rhyme Scheme Agreement	0	0.55	0.77	1
Rhyme Scheme Accuracy	0	0.20	0.60	1
Semantic Similarity	0	0.23	0.62	1
Keyword Similarity	0	0.34	0.64	1
Line-by-line Keyword Similarity	0	0.22	0.45	1
Phoneme Repetition Difference	1	0.12	0.08	0
BLEU-2	0	0.00	0.04	1
chrF	0	0.09	0.16	1

Table 6.1: Random baseline values and Target values of metrics from the evaluation function, measured on the Bilingual Song Lyrics dataset

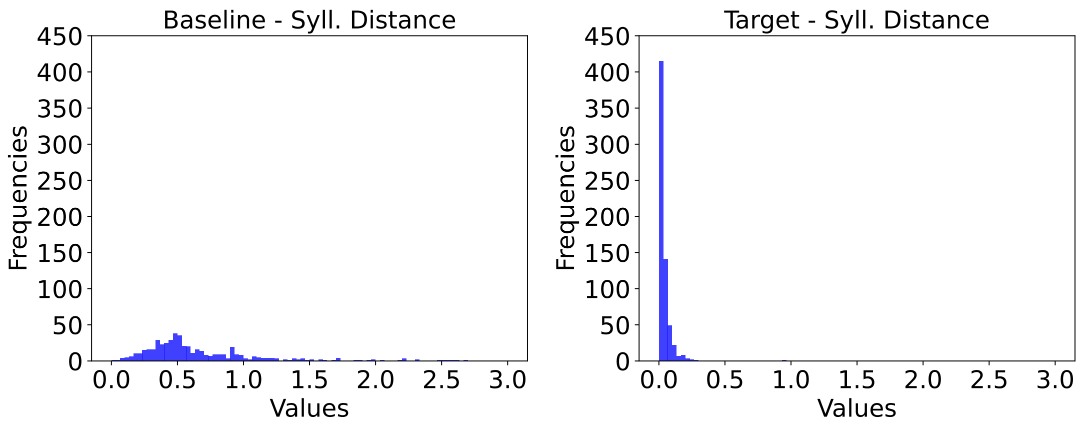


Figure 6.1: Syllable distance of the random baseline and the target

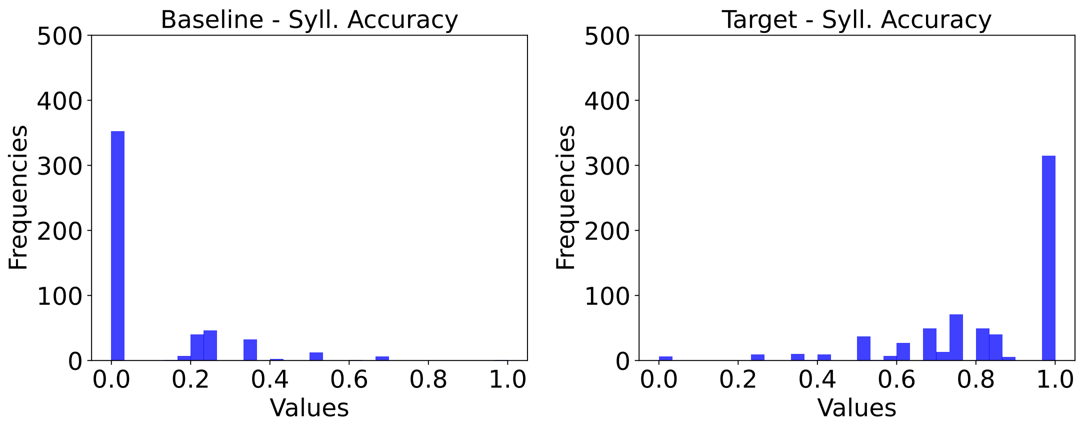


Figure 6.2: syllable accuracy of the random baseline and the target

Random baseline values and target values comparison

In figures 6.1 to 6.10, we can see the specific distributions produced by the evaluation function both on the paired song sections and the randomly shuffled song sections.

Syllable distance is arguably the most important metric out of the evaluation function and Figure 6.1 shows the big difference between syllable distances on paired and shuffled song sections. As we can see in Figure 6.2, for the random baseline, the syllable accuracy is mostly 0, while for the target values the accuracy is oftentimes 1.

We can see both in Figure 6.3 and Figure 6.4, that the target values have a higher concentration of results closer to 1, but not by much. This is because rhyme is very important in song lyrics, so much, so that it is a rarity to find a song section without a rhyme. Also, the length of a song section in terms of lines is limited, therefore there are not that many options in which the two sections could have their rhymes arranged, making the chance of the same rhyme scheme higher. Some rhymes also do not have to be registered by the rhyme detector (see section 5.1.2).

The phoneme repetition difference has a high peak for the low values for the target, while for the random baseline, it has smaller peaks even at high numbers like 0.45 and 0.60 (see Figure 6.12).

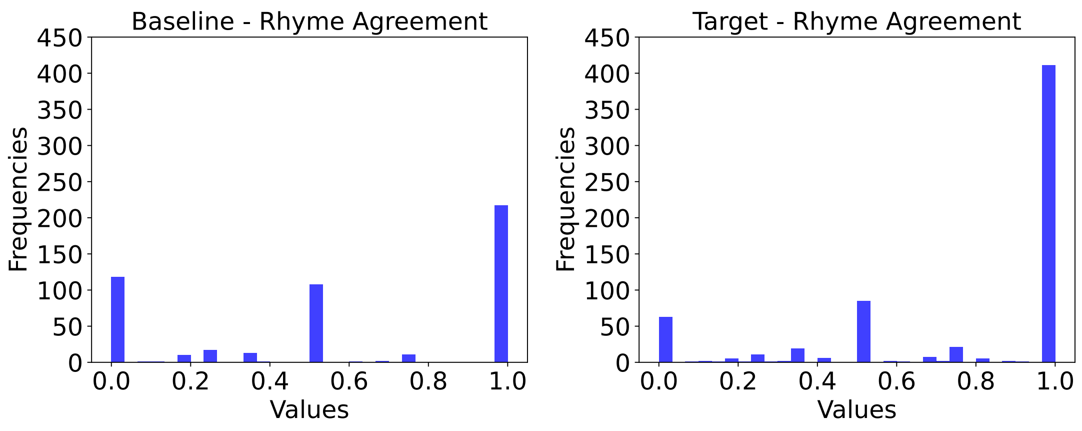


Figure 6.3: Rhyme scheme agreement of the random baseline and the target

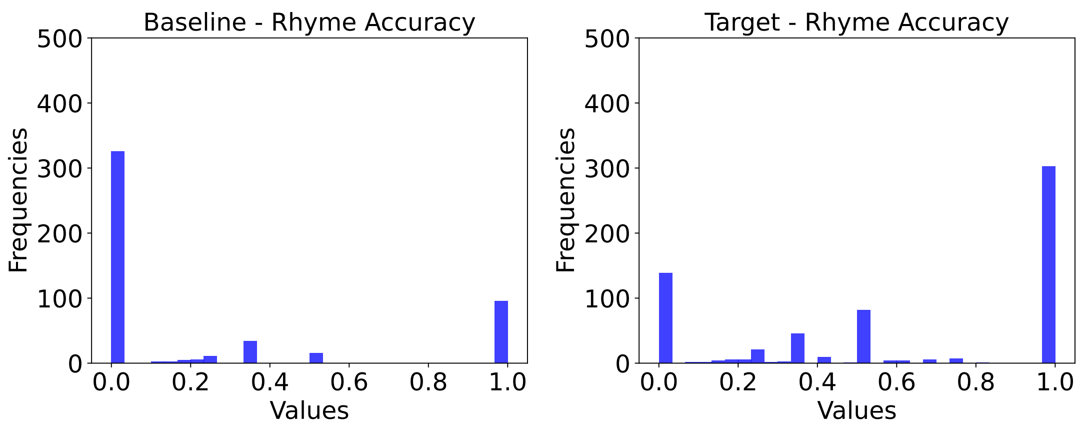


Figure 6.4: Rhyme scheme accuracy of the random baseline and the target

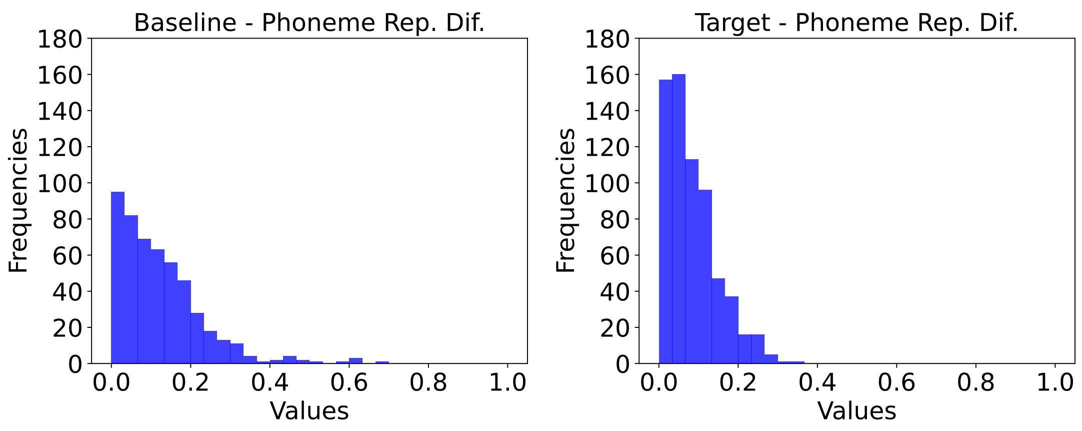


Figure 6.5: Phoneme repetition difference of the random baseline and the target

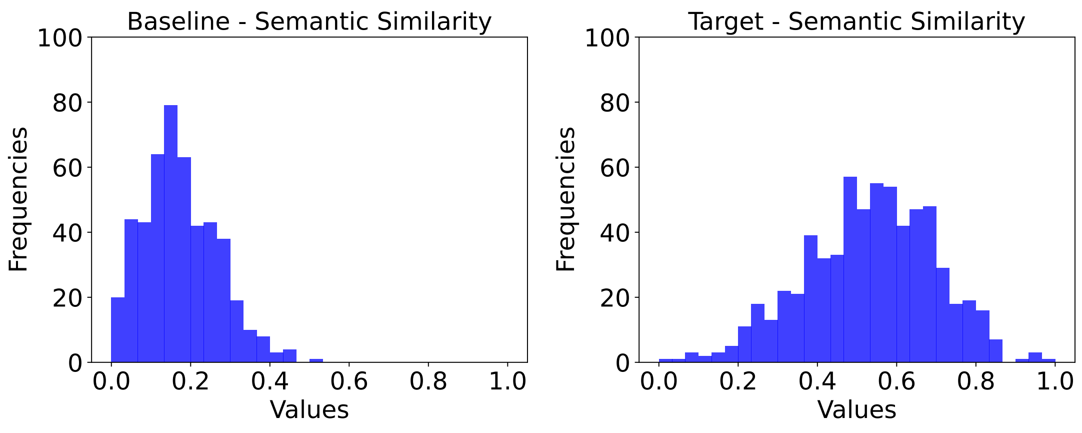


Figure 6.6: Semantic similarity of the random baseline and the target

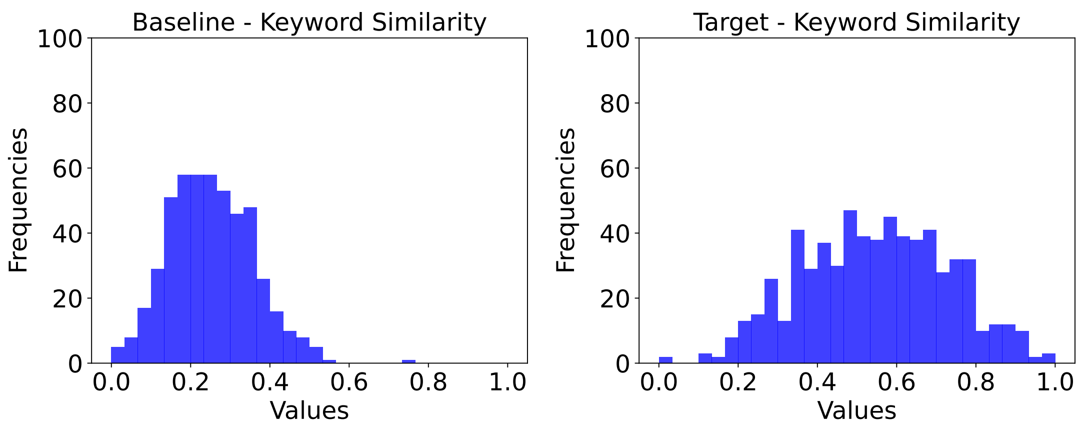


Figure 6.7: Keyword similarity of the random baseline and the target

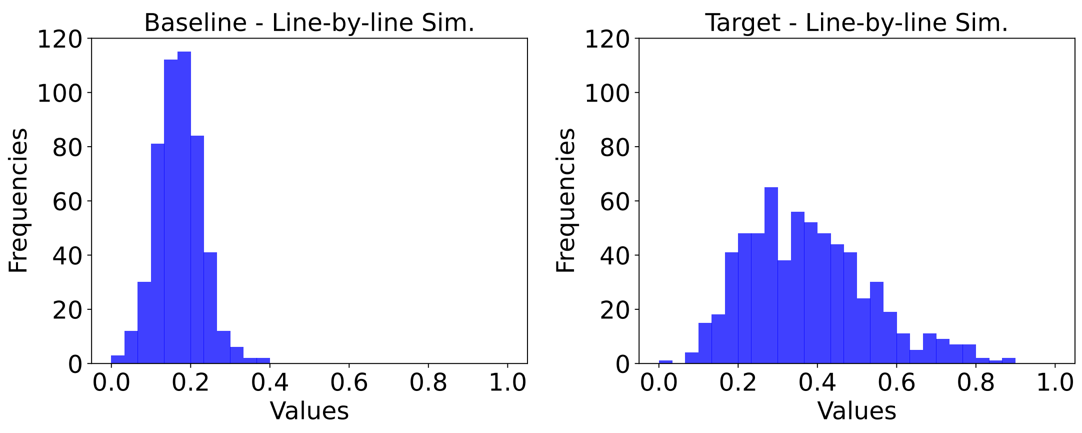


Figure 6.8: Line-by-line keyword similarity of the random baseline and the target

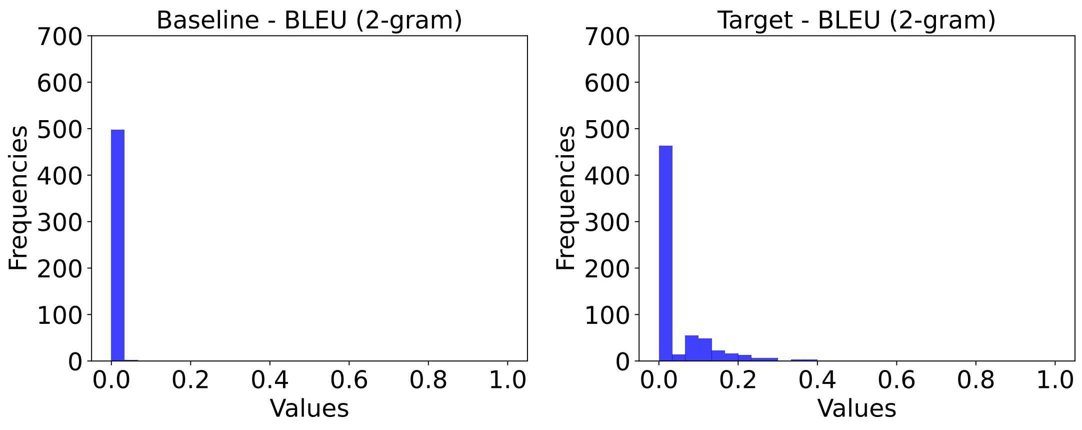


Figure 6.9: BLEU2 score of the random baseline and the target

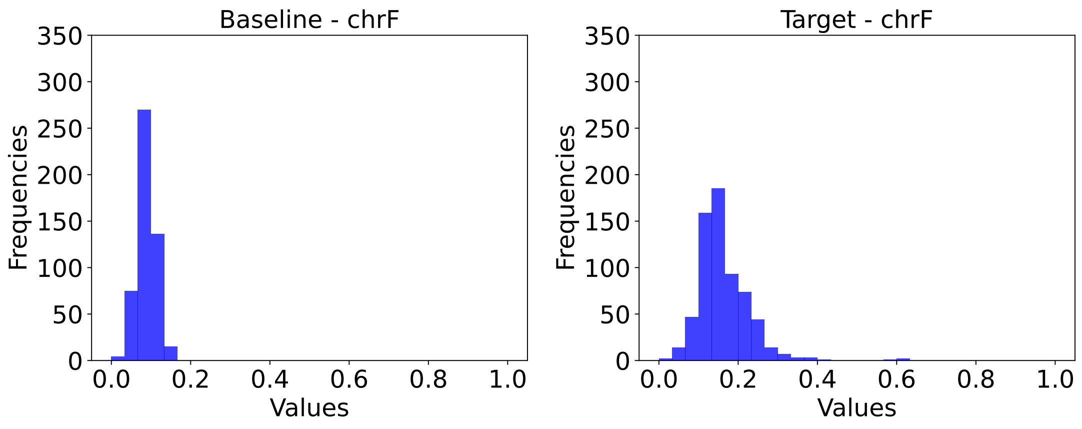


Figure 6.10: chrF score of the random baseline and the target

We can observe a common trend in Figures 6.6, 6.7 and 6.8, where for the baseline, the values lean towards the smaller numbers, while for the target the values tend to be closer to 1. We can also see that the Line-by-line keywords have overall much lower similarity (with averages of 0.22 for baseline and 0.45 for target) than keywords of whole sections (with averages of 0.34 for baseline and 0.64 for target). This might be due to human translators taking the liberty of changing the order or the detail of individual topics of the song sections, as debated by [Kim et al., 2023b].

Lastly, let us look at the translation scores in Figures 6.9 and 6.10, showing the results of the two translation metrics used, BLEU2 and chrF. We observe very low values for both of the translation scores both on the randomly shuffled song sections and on the correctly paired song sections. This proves the point that song translation is not a typical translation task.

6.1.1 Machine translator baseline

In this subsection, we are going to present the results of the evaluation function with an English original and a Czech machine translation (MT) on the input.

We can see in Table 6.2 that although unaltered machine translation is not an ideal way of making lyrics for cover songs, some of the results are quite decent.

	worst	**Random**	**MT**	**Target**	**best**
Syllable Distance	inf	0.65	0.26	**0.03**	0
Syllable Accuracy	0	0.10	0.20	**0.83**	1
Rhyme Scheme Agreement	0	0.55	0.32	**0.77**	1
Rhyme Scheme Accuracy	0	0.20	0.27	**0.60**	1
Semantic Similarity	0	0.23	**0.91**	0.62	1
Keyword Similarity	0	0.34	**0.88**	0.64	1
Line-by-line Keyword Sim.	0	0.22	**0.81**	0.45	1
Phoneme Repetition Dif.	1	0.12	**0.08**	**0.08**	0
BLEU2	0	0.00	**0.49**	0.04	1
chrF	0	0.09	**0.56**	0.16	1

Table 6.2: Machine translator baseline values

The biggest problem with machine translation is large syllable distance and small syllable accuracy. Another problem is the lack of the rhyme scheme and the overall poeticism of the text. Even though the rhyme scheme agreement is quite low (0.32, compared to the baseline value of 0.55), rhyme accuracy is reasonable (0.27, with the baseline being 0.20 and the target being 0.60). This might be due to songs often relying on absolute rhymes (see section 5.1.2). When two lines end with the same word, in languages with similar word order, the translation of both of these lines will probably be the same and, therefore, rhyme again.

Unsurprisingly, all metrics measuring the semantics of the lyrics give high scores, even higher scores than the target values. As was mentioned by Low [2003], the hardest job of a human translator translating the lyrics is to balance the meaning and singability. Machine translation is keeping the meaning well, but the singability is not there.

6.2 Models

In this section, we will describe the individual models we used. The results of the automatic evaluation of outputs of these models can be found in section 6.3, and the results of the manual evaluation of outputs of the best subset of these models can be found in section 6.4.

We experimented with three base models: Mistral7b, TinyLlama and GPT2. For further work, we used models pre-trained on Czech data from the English base models. All of the models we used have knowledge of the Czech Language of varying levels, but never none. Outputs of the pre-trained models before we made any changes given a prompt: "Zapadající slunce" *["The setting sun"]* can be seen in Table 6.4.

We use cswikimistral_0.1[1] pre-trained by Petr Šimeček on the data from Czech Wikipedia. We use two models by Fajčík et al. [2024] from BUT-FIT: CSTinyLlama-1.2B[2] and Czech-GPT-2-XL-133k[3], pre-trained on a Czech data

[1]`https://huggingface.co/simecek/cswikimistral_0.1`
[2]`https://huggingface.co/BUT-FIT/CSTinyLlama-1.2B`
[3]`https://huggingface.co/BUT-FIT/Czech-GPT-2-XL-133k`

Abbr.	Meaning
MIS	cswikimistral_0.1
BTL	CSTinyLlama-1.2B
BGPT2	Czech-GPT-2-XL-133k
OGPT2	Czech-GPT2-OSCAR
TL	TinyLlama-1.1B
WS	trained on whole texts with syllables
WK	trained on whole texts with keywords
WSK	trained on whole texts with syllables and keywords
WSKR	trained on whole texts with syllables, keywords and rhyme schemes
LS	trained on single lines with syllables
LK	trained on single lines with keywords
LSK	trained on single lines with syllables and keywords
LST	trained on single lines with syllables and unrhymed (translated) lines

Table 6.3: Abbreviations used in Sections 6.3 and 6.4

corpus, from which 95.79% is released as *BUT-Large Czech Collection*[4], also by Fajčík et al. [2024]. Then, we used the Czech-GPT2-OSCAR[5] model by Chaloupský [2022] based on the GPT2 small model. And lastly, we used the TinyLlama-1.1B[6] model by [Zhang et al., 2024], which even though is not specifically pre-trained for Czech, from the initial experiments it seems that it has some basic knowledge of the language. Due to our training resources, we chose to fine-tune just the smaller models, meaning CSTinyLlama-1.2B, Czech-GPT-2-XL-133k, Czech-GPT2-OSCAR and TinyLlama-1.1B, as described in sections 5.3.1 and 5.3.2. We tried a few-shot prompting approach, described in section 5.3.3, for the larger model (cswikimistral_0.1). For simplicity of labelling the tables in Sections 6.3, 6.4, we introduce the abbreviations of the models' names and of the prompts they were trained on in Table 6.3.

While fine-tuning the models, we noticed that the performance improved after completing the first epoch of training but did not change further in subsequent epochs. We tested this hypothesis with all the models using a few different training prompts, and all of these experiments did not see any improvement in neither the metrics nor the lyrical quality of the outputs based on our subjective judgement. The results of one of these experiments can be seen in Figure 6.11.

6.3 Automatic evaluation

In this section, we will present the results of the automatic evaluation of both the fine-tuned and the few-shotted models.

[4]`https://huggingface.co/datasets/BUT-FIT/BUT-LCC`

[5]`https://huggingface.co/lchaloupsky/czech-gpt2-oscar`

[6]`https://huggingface.co/TinyLlama/TinyLlama-1.1B-intermediate-step-1431k-3T`

Model	Output
cswikimistral_0.1	Zapadající slunce (v anglickém originále Setting Sun) je epizoda seriálu Star Trek: Vesmírná loď Voyager. V originále se původně vysílala 8. ledna 1998...
CSTinyLlama-1.2B	Zapadající slunce pomalu zapadalo za obzor. I když bylo teprve šest hodin, už nebylo takové vedro. Zabalená v dece jsem se těšila, jak si večer zajdu s pejskem na procházku. Jenže v půl osmé jsem se probudila a začala přemýšlet, co s načatým dnem. Nakonec jsem se rozhodla zajet nakoupit a připravit se na výlet. ”Co si to zase vymýšlíš?” zeptala se mě spolubydlící...
Czech-GPT-2-XL-133k	Zapadající slunce osvětluje město a v dálce se třpytí moře. Když se lidé procházejí v těchto ranních hodinách, většinou si při tom v hlavě přehrávají právě tyto dvě události. ”Ne, je to spíš proto, že je to jeden...
Czech-GPT2-OSCAR	Zapadající slunce na obloze, které se prodíralo na protější stranu od našeho domu, si mne tak získalo. Vzrušovalo mne, že v takovém úleku se nemůže ani pohnout. Spatřil jsem, jak se mu noha, která se ještě pohybovala, třese. Pomalu jsem se posadil. Jak je to možné?“ zeptal jsem se zděšeně...
TinyLlama-1.1B	Zapadající slunce) - původní připojení mezi Słońcem a slunovratem Reference Externí odkazy Selhánky

Table 6.4: Base models outputs, prompted by “Zapadající slunce”

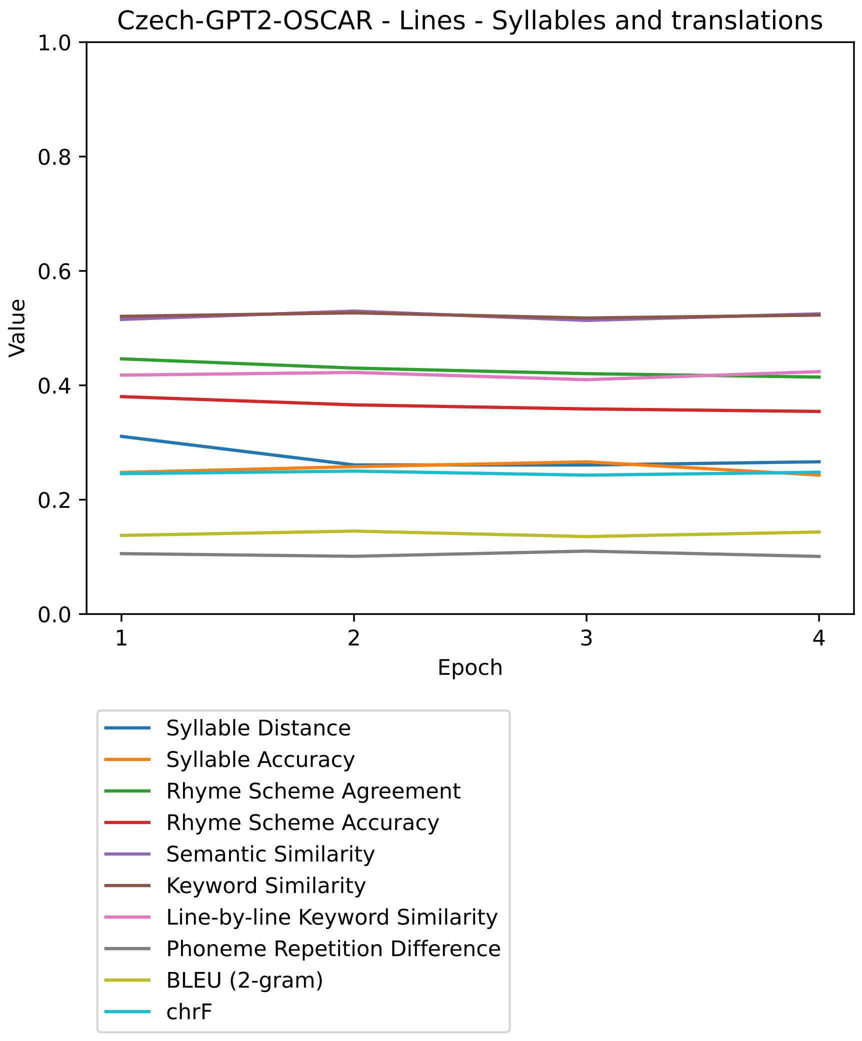


Figure 6.11: The results of individual metrics from evaluation function taken after each epoch of fine-tuning

6.3.1 Raw outputs

First, we will present the results of individual metrics and then show specific examples of generated outputs. All of the following values are measured on the raw output of the model without any postprocessing, other than removing the tags containing information about the desired structure. Based on these results we choose the six interesting models and further evaluate the outputs of these models after various postprocessing functions in Section 6.3.2.

	WS	WK	WSK	WSKR	LS	LK	LSK	LST
Random baseline	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
OGPT2	0.22	0.65	0.23	0.31	0.18	0.61	0.3	0.31
BGPT2	0.26	0.65	0.18	0.49	0.91	1.57	0.49	1.77
BTL	0.18	0.67	0.2	0.17	3.34	0.77	0.31	0.36
TL	0.13	0.75	0.18	0.19	0.09	0.63	0.2	0.17
MIS-5shot	0.69	1.08	0.82	0.78	-	-	-	-
MIS-10shot	0.68	0.91	0.73	0.76	-	-	-	-
Target	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03

Table 6.5: Syllable distance on raw outputs

Let us look at the syllable distance shown in Table 6.5. We can see several trends in the values of the metric. The syllable distance is high when using a WK model (trained on whole texts with keywords) and LK (trained on single lines with keywords). This is not surprising, because these are the only two dataset preprocessing methods where the syllable count was omitted. We can also see that the models trained on single lines are in general performing worse than the models trained on whole texts, except for TL (TinyLlama-1.1B). It is also interesting to note that the models trained on whole texts have a similar syllable distance when being trained on syllables only, syllables and words together, and for the bigger models (CSTinyLlama-1.2B, TinyLlama-1.1B) even on syllables, words and rhyme schemes together. On the other hand, models trained on the lines are not very consistent, again except for TinyLlama-1.1B (TL). The few-shotted Mistral model also has a high syllable distance.

	WS	WK	WSK	WSKR	LS	LK	LSK	LST
Random baseline	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
OGPT2	0.29	0.11	0.28	0.24	0.32	0.11	0.21	0.25
BGPT2	0.25	0.11	0.32	0.14	0.22	0.09	0.18	0.09
BTL	0.35	0.1	0.34	0.34	0.11	0.11	0.27	0.24
TL	0.45	0.1	0.41	0.43	0.57	0.11	0.28	0.37
MIS-5shot	0.12	0.09	0.1	0.1	-	-	-	-
MIS-10shot	0.11	0.1	0.11	0.1	-	-	-	-
Target	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83

Table 6.6: Syllable accuracy on raw outputs

The syllable accuracy shown in Table 6.6 yields similar results to syllable distance (see Table 6.5), showing that this metric performs poorly on models trained on keywords only and on the few-shotted Mistral and that TinyLlama-1.1B reaches quite decent accuracies without any postprocessing.

	WS	WK	WSK	WSKR	LS	LK	LSK	LST
Random baseline	0.55	0.55	0.55	0.55	0.55	0.55	0.55	0.55
OGPT2	0.47	0.48	0.58	0.45	0.61	0.5	0.47	0.45
BGPT2	0.58	0.58	0.52	0.48	0.66	0.45	0.4	0.48
BTL	0.52	0.55	0.57	0.54	0.84	0.37	0.36	0.4
TL	0.42	0.44	0.47	0.47	0.71	0.53	0.62	0.59
MIS-5shot	0.29	0.26	0.33	0.36	-	-	-	-
MIS-10shot	0.36	0.3	0.35	0.39	-	-	-	-
Target	0.77	0.77	0.77	0.77	0.77	0.77	0.77	0.77

Table 6.7: Rhyme scheme agreement on raw outputs

Rhyme scheme agreement values in Table 6.7 show that adding rhyme scheme notation during the preprocessing of the training data (WSKR) has no significant effect compared to WS, WR and WSK, even making it slightly worse. On the other hand, we can see that except for TinyLlama-1.1B and the preprocessing of dataset LS, the results of rhyme scheme agreement are worse for models trained on lines, even though generating the song section line by line should promote rhyme scheme agreement. Furthermore, most of the results stay close to the baseline. As discussed in Section 6.1, due to the rhyming nature of song lyrics, the baseline for rhyming agreement is quite high.

	WS	WK	WSK	WSKR	LS	LK	LSK	LST
Random baseline	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
OGPT2	0.32	0.31	0.22	0.34	0.53	0.43	0.4	0.38
BGPT2	0.21	0.18	0.21	0.26	0.45	0.39	0.36	0.27
BTL	0.21	0.19	0.23	0.29	0.41	0.33	0.33	0.35
TL	0.19	0.18	0.18	0.35	0.69	0.49	0.58	0.55
MIS-5shot	0.17	0.16	0.17	0.19	-	-	-	-
MIS-10shot	0.17	0.16	0.18	0.21	-	-	-	-
Target	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6

Table 6.8: Rhyme scheme accuracy on raw outputs

Even though the rhyme scheme agreement is lower for models trained on single lines (see Table 6.7), Table 6.8 shows that the rhyme scheme accuracy is higher for these models. Also, considering method WSKR, while there was no improvement in rhyme scheme agreement, there is a visible improvement in rhyme scheme accuracy. This could be due to the models without the knowledge of the correct rhyme scheme generating more natural lyrics with more rhymes, while the model with the knowledge of the desired rhyme scheme trying to fill in the desired form, therefore generating fewer rhymes in general.

Semantic similarity (see Table 6.9) is higher on results from models trained on keywords or translated lines than the ones trained purely on syllable counts. Except for the models trained on translated lines, the song sections generated as a whole yield higher semantic accuracy than those generated line by line.

	WS	WK	WSK	WSKR	LS	LK	LSK	LST
Random baseline	0.23	0.23	0.23	0.23	0.23	0.23	0.23	0.23
OGPT2	0.17	0.34	0.38	0.4	0.17	0.33	0.29	0.52
BGPT2	0.17	0.41	0.45	0.46	0.16	0.37	0.41	0.25
BTL	0.16	0.4	0.46	0.47	0.13	0.39	0.37	0.55
TL	0.18	0.37	0.4	0.39	0.2	0.32	0.34	0.41
MIS-5shot	0.18	0.33	0.4	0.42	-	-	-	-
MIS-10shot	0.17	0.39	0.46	0.48	-	-	-	-
Target	0.62	0.62	0.62	0.62	0.62	0.62	0.62	0.62

Table 6.9: Semantic similarity on raw outputs

	WS	WK	WSK	WSKR	LS	LK	LSK	LST
Random baseline	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34
OGPT2	0.22	0.44	0.49	0.53	0.24	0.42	0.36	0.52
BGPT2	0.23	0.53	0.58	0.62	0.18	0.44	0.5	0.28
BTL	0.22	0.52	0.6	0.61	0.15	0.46	0.46	0.58
TL	0.25	0.49	0.53	0.52	0.27	0.39	0.42	0.45
MIS-5shot	0.26	0.45	0.54	0.56	-	-	-	-
MIS-10shot	0.23	0.51	0.6	0.64	-	-	-	-
Target	0.64	0.64	0.64	0.64	0.64	0.64	0.64	0.64

Table 6.10: Keyword similarity on raw outputs

Keyword similarity (see Table 6.10) yields similar results as semantic similarity (see Table 6.9) in terms of inter-model comparison. As opposed to semantic similarity, where the results are average (considering the baseline for semantic similarity is 0.23 and the target is 0.6), The results of keyword similarity, especially those of BGPT2 and BTL when generating whole texts, are quite satisfactory (considering the baseline value is 0.34 and the target value is 0.64). Also, while the few-shotted Mistral was below average in metrics measuring rhyme and syllable counts, with just 10 examples in the prompt of the WSKR prompt type it reached the target value.

	WS	WK	WSK	WSKR	LS	LK	LSK	LST
Random baseline	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22
OGPT2	0.17	0.24	0.25	0.27	0.17	0.33	0.28	0.42
BGPT2	0.17	0.25	0.26	0.27	0.17	0.39	0.41	0.16
BTL	0.16	0.26	0.28	0.28	0.17	0.39	0.37	0.44
TL	0.17	0.24	0.26	0.24	0.17	0.32	0.36	0.35
MIS-5shot	0.16	0.19	0.24	0.26	-	-	-	-
MIS-10shot	0.16	0.22	0.26	0.27	-	-	-	-
Target	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45

Table 6.11: Line-by-line keyword similarity on raw outputs

Line-by-line keyword similarity on the other hand unsurprisingly yields better results for the models generating song sections line by line, as can be seen in Table 6.11.

	WS	WK	WSK	WSKR	LS	LK	LSK	LST
Random baseline	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12
OGPT2	0.22	0.23	0.2	0.27	0.11	0.1	0.13	0.11
BGPT2	0.23	0.22	0.19	0.17	0.38	0.19	0.11	0.14
BTL	0.2	0.21	0.21	0.17	0.64	0.13	0.14	0.13
TL	0.15	0.15	0.17	0.13	0.1	0.11	0.11	0.1
MIS-5shot	0.14	0.14	0.15	0.14	-	-	-	-
MIS-10shot	0.15	0.14	0.15	0.14	-	-	-	-
Target	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08

Table 6.12: Phoneme repetition distance on raw outputs

Phoneme repetition distance (see Table 6.12) is generally smaller when the song section is generated line by line. This might be due to language models often getting caught in repeating the same section of a text over and over, and generating each line separately breaks this cycle.

	WS	WK	WSK	WSKR	LS	LK	LSK	LST
Random baseline	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
OGPT2	0.0	0.0	0.01	0.01	0.0	0.0	0.0	0.14
BGPT2	0.0	0.01	0.01	0.01	0.0	0.0	0.01	0.03
BTL	0.0	0.01	0.01	0.01	0.0	0.01	0.0	0.13
TL	0.0	0.0	0.01	0.01	0.0	0.0	0.0	0.11
MIS-5shot	0.0	0.0	0.0	0.0	-	-	-	-
MIS-10shot	0.0	0.0	0.01	0.0	-	-	-	-
Target	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04

Table 6.13: BLEU-2 on raw outputs

As already stated in Section 6.1, BLEU is not an ideal metric for our problem, therefore the values are low, regardless of the model or the prompt format, except for the models trained on the translated lines. The BLEU score of these outputs is still significantly smaller than the BLEU score of the machine-translated outputs, achieving 0.49 (see Section 6.1.1), compared to the highest score from the models being 0.14.

	WS	WK	WSK	WSKR	LS	LK	LSK	LST
Random baseline	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09
OGPT2	0.09	0.1	0.11	0.11	0.08	0.11	0.1	0.25
BGPT2	0.09	0.11	0.13	0.14	0.05	0.1	0.12	0.09
BTL	0.09	0.11	0.12	0.13	0.02	0.12	0.12	0.23
TL	0.09	0.1	0.13	0.12	0.09	0.11	0.12	0.22
MIS-5shot	0.09	0.1	0.11	0.11	-	-	-	-
MIS-10shot	0.08	0.11	0.12	0.12	-	-	-	-
Target	0.16	0.16	0.16	0.16	0.16	0.16	0.16	0.16

Table 6.14: chrF on raw outputs

As we can see in Table 6.14, chrF follows a similar trend as BLEU, but slight improvements in the score can be seen in places where semantic similarity and keyword similarity were higher.

As we can see in Table 6.4, TinyLama-1.1B (TL) can not generate Czech very well as it is making up new words. On the other hand, the following tables show it fulfils the formal requirements with ease. That is mainly due to disrespecting the naturalness of the language. From this, an intriguing conclusion can be drawn: The models having more difficulty obeying the formal requirements should generate rather natural Czech texts. It is more difficult to satisfy the formal requirements while generating valid sentences than while ignoring the language altogether.

Choosing the best models

In the end, we decided to proceed with the four following models: CSTinyLlama-1.2B trained on whole texts with specified syllables, keywords and rhyme schemes (BTL-WSKR), Czech-GPT-2-XL-133k trained on whole texts with specified syllables and keywords (BGPT2-WSK), TinyLlama-1.1B trained on single lines with specified syllables and keywords (TL-LSK) and finally CSTinyLlama-1.2B trained on single lines with specified syllables and translated English original lyrics (BTL-LST).

We chose these four models by comparing the automatic evaluation scores and by trying to bring the diversity of the models to the next phase. Out of the models generating whole song sections at once, BTL-WSKR had the most stable performance across all the metrics. Although the BGPT2 model sometimes had problems with straying away from the form of the English input section, the BGPT2-WSK model specifically achieved relatively good values on the evaluation metrics. The TL model did overall great on the evaluation metrics and we chose TL-LSK for the postprocessing evaluation. BTL-LST was chosen mainly to assess the effect of putting translated English input into the prompt and see if the quality of the generated song sections improves with the postprocessing.

We also chose to evaluate the effects of the postprocessing functions on the Czech-GPT2-OSCAR trained on whole texts with marked syllables and keywords (OGPT2-WSK), because this model is small enough to run locally, while still producing acceptable results, and on a 10-shot prompted cswikimistral_0.1 (MIS-10shot) to see whether the postprocessing could be enough to combat the lack of fine-tuning.

6.3.2 After postprocessing

In this section, we are going to show the results of the individual metrics of the evaluation function after applying various postprocessing methods to the generated song section. We are going to compare the individual postprocessing functions and their combinations, as well as the performance of individual models. We will also present example outputs of these models with various postprocessing functions applied. The two postprocessing functions we are experimenting with are choosing the best out of n generated outputs (see Section 5.4.1) and removing or adding stopwords to achieve a better syllable distance (see Section 5.4.2).

In the following tables, we show the results of the random baseline, then the raw output of the model, then the *choose best* postprocessing function with n

generated song sections to choose from equal to 5 and 10. After that, we show the raw output with the *stopwords* function applied, as well as combining the two postprocessing functions and showing the results after applying both of them to the raw model output. Last, we show the target values measured on the Human-made cover songs.

	BTL WSKR	BGPT2 WSK	OGPT2 WSK	TL LSK	BTL LST	F-MIS WSKR
Random baseline	0.65	0.65	0.65	0.65	0.65	0.65
Raw output	0.17	0.18	0.23	0.2	0.17	0.76
Choose from 5	0.11	0.12	0.15	0.05	0.07	0.55
Choose from 10	0.1	0.11	0.13	0.02	0.04	0.5
Stopwords	0.03	0.03	0.06	0.02	0.07	0.17
Stopwords + Choose from 5	0.01	0.01	0.01	0.0	0.0	0.06
Stopwords + Choose from 10	0.0	0.01	0.01	0.0	0.0	0.04
Target	0.03	0.03	0.03	0.03	0.03	0.03

Table 6.15: Syllable distance after postprocessing

As we can see in Table 6.15, The syllable distance improved drastically with the postprocessing functions, which is not surprising because the *choose best* postprocessing is choosing syllable distance as one of its parameters, and *stopwords* postprocessing is based purely on minimizing the syllable distance between the English original and the Czech generated cover version. In this case, when choosing the best out of 5 or 10 versions with stopwords added or removed, the syllable distance reaches even better values than the target data.

	BTL WSKR	BGPT2 WSK	OGPT2 WSK	TL LSK	BTL LST	F-MIS WSKR
Random baseline	0.1	0.1	0.1	0.1	0.1	0.1
Raw output	0.34	0.32	0.28	0.28	0.37	0.1
Choose from 5	0.5	0.44	0.4	0.73	0.63	0.16
Choose from 10	0.53	0.5	0.46	0.89	0.79	0.2
Stopwords	0.94	0.93	0.92	0.93	0.91	0.82
Stopwords + Choose from 5	0.97	0.97	0.97	1.0	0.99	0.93
Stopwords + Choose from 10	0.98	0.98	0.98	1.0	1.0	0.95
Target	0.83	0.83	0.83	0.83	0.83	0.83

Table 6.16: Syllable accuracy after postprocessing

The same trend can be seen in Table 6.16, showing the syllable accuracy. We can see that the *choose best* function improves the syllable accuracy quite a bit, eliminating the outliers dragging the score down, but the true difference is made by the *stopwords* postprocessing function, which makes the output of the model score a lot better than the actual target syllable accuracy.

Rhyme agreement is slightly improving with the additional postprocessing functions (see Table 6.17). This is interesting because the *stopwords* postprocessing does not directly manipulate the rhyme scheme, but we can see that the rhyme scheme is influenced by the fact whether the *stopwords* technique is used or not. This is due to *stopwords* correcting the number of syllables in each line,

	BTL WSKR	BGPT2 WSK	OGPT2 WSK	TL LSK	BTL LST	F-MIS WSKR
Random baseline	0.55	0.55	0.55	0.55	0.55	0.55
Raw output	0.54	0.52	0.49	0.62	0.59	0.39
Choose from 5	0.57	0.51	0.56	0.68	0.46	0.42
Choose from 10	0.58	0.52	0.57	0.71	0.53	0.45
Stopwords	0.56	0.51	0.5	0.63	0.4	0.38
Stopwords + Choose from 5	0.62	0.54	0.58	0.73	0.53	0.44
Stopwords + Choose from 10	0.64	0.59	0.62	0.74	0.61	0.49
Target	0.77	0.77	0.77	0.77	0.77	0.77

Table 6.17: Rhyme scheme agreement after postprocessing

therefore the *choose best* function can decide which section is the best based on the other metrics than syllable distance.

	BTL WSKR	BGPT2 WSK	OGPT2 WSK	TL LSK	BTL LST	F-MIS WSKR
Random baseline	0.2	0.2	0.2	0.2	0.2	0.2
Raw output	0.29	0.21	0.3	0.58	0.55	0.21
Choose from 5	0.36	0.27	0.29	0.64	0.42	0.23
Choose from 10	0.4	0.3	0.28	0.67	0.49	0.26
Stopwords	0.29	0.21	0.29	0.6	0.35	0.2
Stopwords + Choose from 5	0.42	0.29	0.27	0.69	0.5	0.26
Stopwords + Choose from 10	0.42	0.35	0.33	0.71	0.57	0.3
Target	0.6	0.6	0.6	0.6	0.6	0.6

Table 6.18: Rhyme scheme accuracy after postprocessing

Similarly as with rhyme scheme agreement, rhyme scheme accuracy (see Table 6.18) is improving with the added postprocessing functions. The rhyme scheme accuracy is even higher than the target value on the outputs from the TinyLlama-1.1B model trained on individual lines with syllables and keywords.

	BTL WSKR	BGPT2 WSK	OGPT2 WSK	TL LSK	BTL LST	F-MIS WSKR
Random baseline	0.23	0.23	0.23	0.23	0.23	0.23
Raw output	0.47	0.45	0.38	0.34	0.41	0.48
Choose from 5	0.51	0.49	0.42	0.4	0.64	0.51
Choose from 10	0.53	0.51	0.43	0.42	0.65	0.52
Stopwords	0.46	0.44	0.37	0.35	0.53	0.45
Stopwords + Choose from 5	0.53	0.5	0.44	0.47	0.67	0.49
Stopwords + Choose from 10	0.54	0.53	0.46	0.51	0.7	0.51
Target	0.62	0.62	0.62	0.62	0.62	0.62

Table 6.19: Semantic similarity after postprocessing

As with the rhyme accuracy and rhyme agreement, Table 6.19 shows improvement of semantic similarity with additional postprocessing functions. The semantic similarity of the outputs in general is well over the baseline, and the semantic similarity of the model trained on syllables and translated lines even surpasses the target.

	BTL WSKR	BGPT2 WSK	OGPT2 WSK	TL LSK	BTL LST	F-MIS WSKR
Random baseline	0.34	0.34	0.34	0.34	0.34	0.34
Raw output	0.61	0.58	0.49	0.42	0.45	0.64
Choose from 5	0.66	0.63	0.53	0.47	0.66	0.67
Choose from 10	0.67	0.65	0.54	0.51	0.67	0.69
Stopwords	0.6	0.58	0.48	0.43	0.56	0.61
Stopwords + Choose from 5	0.67	0.65	0.55	0.54	0.69	0.66
Stopwords + Choose from 10	0.68	0.66	0.57	0.59	0.72	0.69
Target	0.64	0.64	0.64	0.64	0.64	0.64

Table 6.20: Keyword similarity after postprocessing

Keyword similarity of the post-processed outputs is also better than the baseline, as can be seen in Table 6.20. The keyword similarity of the best out of 10 texts, regardless of whether the *stopwords* function is applied, in most cases comfortably reaches the level of the target values. We can see that the *stopwords* function doesn't improve the semantic similarity when applied to the raw output alone, but when applied before the *choose best* function, the results are slightly better than just applying *choose best* function alone. This is again due to *stopwords* minimizing the syllable distance, therefore taking the syllable distance out of the equation when the *choose best* function weights which song section is the best.

	BTL WSKR	BGPT2 WSK	OGPT2 WSK	TL LSK	BTL LST	F-MIS WSKR
Random baseline	0.22	0.22	0.22	0.22	0.22	0.22
Raw output	0.28	0.26	0.25	0.36	0.35	0.27
Choose from 5	0.29	0.27	0.25	0.42	0.54	0.29
Choose from 10	0.29	0.27	0.26	0.47	0.56	0.29
Stopwords	0.28	0.26	0.25	0.35	0.44	0.27
Stopwords + Choose from 5	0.29	0.27	0.26	0.51	0.58	0.29
Stopwords + Choose from 10	0.29	0.27	0.26	0.57	0.62	0.28
Target	0.45	0.45	0.45	0.45	0.45	0.45

Table 6.21: Line-by-line keyword similarity after postprocessing

With line keywords similarity (see Table 6.21) when generating whole song sections, even after the postprocessing, the values are between the baseline and the target. On the other hand, when generating the song sections line by line, the line keyword similarity of outputs after postprocessing is higher than the target values. This shows that human translators and cover writers take the liberty of switching the order of topics within a section.

Phoneme repetition difference (see Table 6.22) is starting much worse than the baseline value, but with the postprocessing getting closer to the target. It is directly influenced by the *choose best* postprocessing function, as *phoneme repetition difference* is one of the four deciding metrics.

The BLEU2 score is not an ideal metric for measuring the quality of song lyrics and Table 6.23 shows it again. Here, we can see that in the case of the model being trained on the translated lines (LST), the BLEU score is better, but while

	BTL WSKR	BGPT2 WSK	OGPT2 WSK	TL LSK	BTL LST	F-MIS WSKR
Random baseline	0.12	0.12	0.12	0.12	0.12	0.12
Raw output	0.17	0.19	0.2	0.11	0.1	0.14
Choose from 5	0.1	0.11	0.12	0.1	0.1	0.12
Choose from 10	0.08	0.09	0.1	0.1	0.1	0.11
Stopwords	0.17	0.18	0.18	0.1	0.12	0.17
Stopwords + Choose from 5	0.08	0.08	0.09	0.09	0.09	0.14
Stopwords + Choose from 10	0.06	0.07	0.07	0.09	0.09	0.12
Target	0.08	0.08	0.08	0.08	0.08	0.08

Table 6.22: Phoneme repetition distance after postprocessing

	BTL WSKR	BGPT2 WSK	OGPT2 WSK	TL LSK	BTL LST	F-MIS WSKR
Random baseline	0.0	0.0	0.0	0.0	0.0	0.0
Raw output	0.01	0.01	0.01	0.0	0.11	0.0
Choose from 5	0.01	0.01	0.01	0.01	0.17	0.01
Choose from 10	0.01	0.01	0.01	0.01	0.17	0.01
Stopwords	0.01	0.01	0.01	0.0	0.1	0.0
Stopwords + Choose from 5	0.01	0.01	0.01	0.01	0.15	0.01
Stopwords + Choose from 10	0.01	0.01	0.01	0.01	0.16	0.01
Target	0.04	0.04	0.04	0.04	0.04	0.04

Table 6.23: BLEU2 after postprocessing

	BTL WSKR	BGPT2 WSK	OGPT2 WSK	TL LSK	BTL LST	F-MIS WSKR
Random baseline	0.09	0.09	0.09	0.09	0.09	0.09
Raw output	0.13	0.13	0.11	0.12	0.22	0.12
Choose from 5	0.14	0.13	0.12	0.13	0.28	0.13
Choose from 10	0.14	0.14	0.12	0.14	0.28	0.13
Stopwords	0.13	0.13	0.11	0.13	0.22	0.12
Stopwords + Choose from 5	0.14	0.14	0.13	0.14	0.27	0.13
Stopwords + Choose from 10	0.14	0.14	0.13	0.14	0.28	0.13
Target	0.16	0.16	0.16	0.16	0.16	0.16

Table 6.24: chrF after postprocessing

choosing the best section out of n improves the quality of the 'translation', adding and removing stopwords slightly lowers the score. This is due to the *stopwords* function changing the word n-grams, which the BLEU score is dependent on.

The results of chrF (Table 6.24) are quite similar to BLEU (Table 6.23), except for chrF improving slightly with the *choose best* postprocessing method. This might be due to two reasons, either the *choose best* method eliminates the outputs that are outliers and neither make sense nor hold any form, or it is due to *choose best* method considering semantic similarity while choosing the best song section.

Next, we will present some of the postprocessed outputs from each model for illustration. Both *stopwords* and *choose best* postprocessing functions were used, with $n = 10$ for the *choose best* function. On the left of each figure, we show the

English input and on the right the model output.

The outputs of BTL-WSKR can be found in Figure 6.12, demonstrating a nice rhyme scheme agreement without unnecessary line repetition, as well as a low syllable distance, and especially in the second section, semantic similarity. Figure 6.13 shows the outputs of BGPT2-WSK, where is a demonstration of an incorrectly formatted English input, which resulted in incorrect keywords extraction, and in this case the model's assumption that "Diff" is a name. Figure 6.14 shows the outputs of OGPT2-WSK. In Figure 6.15 we can see the outputs of TL-LSK, demonstrating an accurate rhyme scheme and perfect syllable accuracies, oftentimes due to the model making up new words to match the structure (e.g. *'staví'* and *'neuvaví'*). The same can be said for BTL-LST outputs in Figure 6.16 (e.g. *'na světle'* and *'v noci tle'*). The Mistral model, while keeping the keywords well, does not follow the syllable counts and rhyme scheme very closely. We can see in Figure 6.17 that the 'add stopwords' postprocessing was used several times, by adding the 'na na na' in the beginning of most lines.

Coming back to [Low, 2003] and his pentathlon of singable translations, automatic evaluation is insufficient for evaluating the aspects of naturalness and singability. For that, manual evaluation is needed.

I'll escape now from that world	*že po všem co se stalo*
From the world of Jean Valjean	*a tak valjean je můj svět*
There is nowhere I can turn	*no tak už se nebojím*
There is no way to go on	*dál jen tam se vrátím zpět*
When the beating of your heart	*kdo to tluče a duní*
Echoes the beating of the drums	*to srdce v rytmu bubnů zní*
There is a life about to start	*zítřejším jednání*
When tomorrow comes	*se na srdce bude hrát*

Figure 6.12: Generated Czech covers of the English original song sections by the BTL-WSKR model with *choose best* of 10 and *stopwords* postprocessing.

So our diff 'rences ain't	*diff je optimista*
really that extreme	*ale sadista*
We're one big team	*jsem extrémní*
Call us brutal Sick Sadistic	*a tak to má být to nevadí*
And grotesquely optimistic	*s tím už se musíme smířit*

Figure 6.13: Generated Czech cover of the English original song section by the BGPT2-WSK model with *choose best* of 10 and *stopwords* postprocessing.

The snow glows white on the mountain tonight	*když padá sníh a hory houkají*
Not a footprint to be seen	*vítr všude kolem nás*
A kingdom of isolation	*a hory k tobě přitulí*
And it looks like I'm the queen	*vítr všude kolem nás*
The wind is howling	*dál a ty nechceš*
like this swirling storm inside	*tak aby se mnou chtěl jít*
Couldn't keep it in heaven knows I tried	*no a já teď chtěl jenom něco říct*

Figure 6.14: Generated Czech cover of the English original song section by the OGPT2-WSK model with *choose best* of 10 and *stopwords* postprocessing.

Do you want to build a snowman	*a ťu přišel stát dělej sníh*
Come on let's go and play	*no tak ťápe si hrát*
I never see you anymore	*copak ťáp a mě zase už*
Come out the door	*tak ďábel už*
It's like you've gone away	*ďábel přijít pryč brát*
We used to be best buddies	*ňák nejlepších kámuši*
And now we're not	*a ťu co nás*
I wish you would tell me why	*že ťápám za každý den*
Do you want to buid a snowman	*ňákej chce sníh už tě staví*
It doesn't have to be a snowman	*ňáký sněhulák neuvaví*

Figure 6.15: Generated Czech cover of the English original song section by the TL-LSK model with *choose best* of 10 and *stopwords* postprocessing.

Ah yes the virtuous Fantine	*že óóc tyctiž ctnostný fantine*
Who keeps herself so pure and clean	*rát se v ní tak čisté je ne*
You'd be the cause I had no doubt	*tak ťa bych nebyl příčinou*
Of any trouble hereabout	*ňák o problémech neslibnou*
You play a virgin in the light	*ťa hraju pannu na světle*
But need no urgin' in the night	*rát není třeba v noci tle*

Figure 6.16: Generated Czech cover of the English original song section by the BTL-LST model with *choose best* of 10 and *stopwords* postprocessing.

And so I read a book	*jsem ona obraz jsem*
Or maybe two or three	*život kniha galerie je*
I'll add a few new paintings to my gallery	*na na na na na na na na na začátek*
I'll play guitar and knit	*na na na začátek*
And cook and basically	*na na na začátek*
Just wonder when will my life begin	*na na na na na na začátek*

Figure 6.17: Generated Czech cover of the English original song section by the Mistral model with 10-shot prompting, *choose best* of 10 and *stopwords* postprocessing.

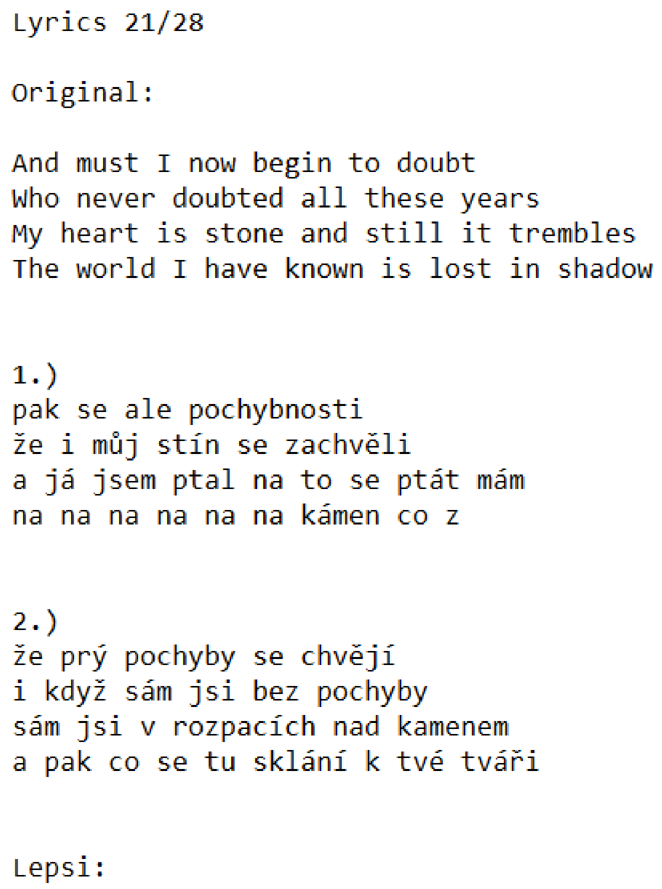
6.4 Manual Evaluation

In this section, we will explain the form of manual evaluation we use, we will present the results and draw conclusions based on these results. The main goal of the manual evaluation is to assess the overall fitness of the lyrics to be the lyrics of a cover song, with a focus on the aspect of naturalness and singability, as these two are not thoroughly covered by the automatic evaluation metrics.

6.4.1 The method

The manual evaluation was done on the output song sections from the same six models that were chosen for postprocessing based on the automatic evaluation metrics (see Section 6.3.1). The sections were generated with the *choose best out of 10* and *stopwords* postprocessing functions. In addition to these six types of generated song sections, two more types were added: the original Czech human-translated covers (denoted as 'cs_lyrics') and the machine translation of the English original lyrics done by the Lindat translator.

We have eight different models and we want to compare each with each to conclude which model yields the best results. We decided to carry out the evaluation using blind pairwise binary comparisons with ties disallowed (A/B testing), providing the evaluators with a pair of outputs of two different systems for the same input and asking them to select which of the two outputs is better than the other. If each inter-model comparison would occur just once, that still means 28 comparisons in total, which is already quite a lot for a participant to fill in. If we created only a single questionnaire of 28 comparisons identical for all of the evaluators, the results would not be very objective, as each model would have just one chance to generate an output for each inter-model comparison. If one model was lucky and the other one not, then the participants would rate the luck of the model, not the actual quality of its average results. As presenting more examples to one participant would be too demanding as well as time-consuming for the participants, we decided to randomly generate a questionnaire containing different output sections for each participant independently. For each inter-model comparison, we randomly picked a song section from the Bilingual Human-translated Lyrics dataset and provided versions of this song section generated by the currently compared models.



```
Lyrics 21/28

Original:

And must I now begin to doubt
Who never doubted all these years
My heart is stone and still it trembles
The world I have known is lost in shadow


1.)
pak se ale pochybnosti
že i můj stín se zachvěli
a já jsem ptal na to se ptát mám
na na na na na na kámen co z


2.)
že prý pochyby se chvějí
i když sám jsi bez pochyby
sám jsi v rozpacích nad kamenem
a pak co se tu sklání k tvé tváři


Lepsi:
```

Figure 6.18: An example of comparison of BTL-WSKR output section and BGPT2-WSK output section, with the prompt ‘Lepsi:’ expecting the id of the better song section.

6.4.2 The questionnaire

The participants were asked to choose the better cover version based on the English original. In the instructions of the questionnaire, we had three criteria in order of priority. First, the better cover should be singable to the same melody as the English original: if the participant does not know the melody of the original, they should concentrate on the rhythmicity of the English original vs the Czech cover. Second, the better cover should be natural, with the words easily following one another. Third, the better cover should sound like song lyrics: it is foremost singable, and it does not have to have exactly the same meaning as the original. An example of one comparison can be seen in Figure 6.18

6.4.3 Results

First, we will discuss the score interpretation. Each time a participant votes for a model in the inter-model comparison, the victorious model gets a point. As the models are 8 in total, each model can obtain a maximum of 7 points per questionnaire, if it is chosen every time. A model can obtain a minimum of 0 points per questionnaire if it does not get chosen at all. For example, when a model has 5 points, it means that it was victorious in 5 inter-model comparisons

source	points out of 7	chosen in % of cases
cs_lyrics	**6.04**	**86**
translation	4.96	71
btl_wskr	**4.00**	**57**
bgpt2_wsk	3.92	56
ogpt2_wsk	3.36	48
btl_lst	2.44	35
mistral	1.76	25
tl_lsk	1.52	22

Table 6.25: Averaged results of the manual evaluation from all 25 participants.

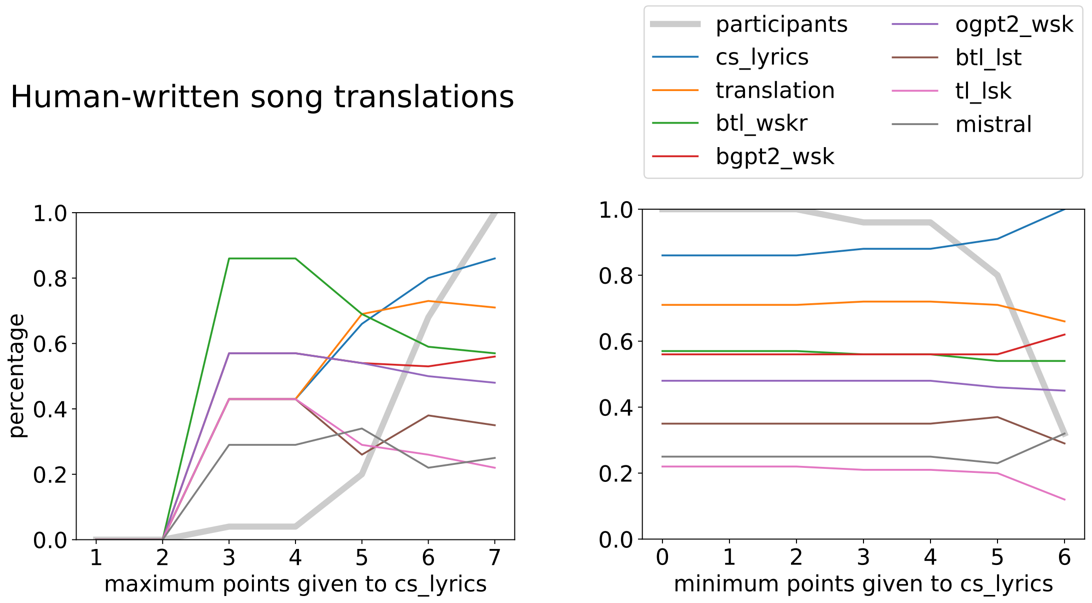
and was defeated in two inter-model comparisons. We can also express the score by the percentage of cases in which a specific model won (computed as the number of obtained points divided by 7 for each questionnaire).

Table 6.25 shows the averaged results of all 25 participants. We can see that the human-translated Czech lyrics (cs_lyrics) are rated the best, with an average rating of 6.04 points and a chance of being chosen of 86%. The second spot in the averaged ratings is taken by the machine translations with the chance of being chosen equal to 71%. Third in order of preference of the participant, is the BUT TinyLlama model fine-tuned on whole sections with syllables, keywords and a rhyme scheme (btl_wskr) with being chosen on average 4 times out of 7, meaning with a 57% chance. All three language models fine-tuned on the whole song sections yield similar results. A drop in the score occurs with the models fine-tuned to generate song sections line by line and on the few-shotted Mistral model.

While conducting the manual evaluation, we noticed that the responses differed from one another, so we decided to further investigate the results. We filter the results by the number of points obtained by a specific model: For example, if we filter the responses by a criterion that the cs_lyrics have to have 6 or 7 points, we get 80% of the responses back. By averaging these responses, we get a representation of which models are preferred by participants who like the human-translated Czech lyrics. On the other hand, if we filter the responses by a criterion that the cs_lyrics have to be less than or equal to 3 points, we get only one response back, showing that not liking the human-translated Czech lyrics is an anomaly. We will explore the results by filtering them in the following figures.

In Figure 6.19, we can see that the score of the human-translated song lyrics (cs_lyrics) is generally high. In the subfigure 6.19a we can see that cs_lyrics getting a low score is an anomaly, as only 4% of participants (one participant) chose the cs_lyrics less than or equal to 4 times. Even though the cs_lyrics are popular, as we can see in the subfigure 6.19b, only 32% of participants gave the cs_lyrics full 7 points (they chose the human-translated song lyrics as the better cover version in every inter-model comparison). This implies that in the results of the other 68% of the participants, the human-translated song lyrics were beaten at least once.

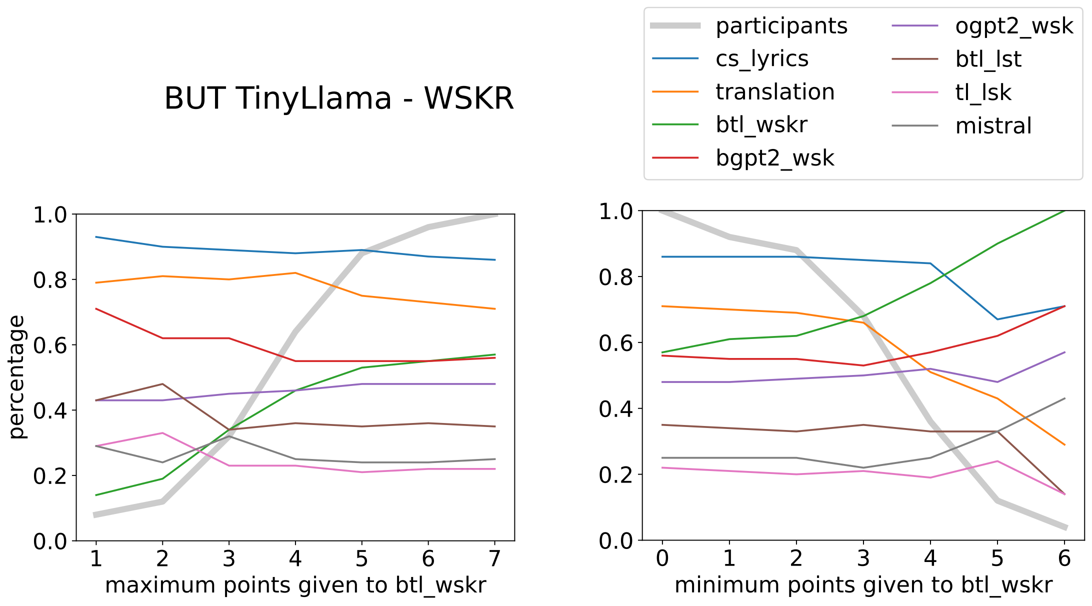
Next, we filter the results by the number of times our best-performing model (btl_wskr) was chosen. The graphs can be seen in Figure 6.20. We can see that the number of people who like the model is roughly the same amount as the



(a) In each tick x on the X-axis, we show the results from participants who gave **less than or equal to** x points to the cs_lyrics model. For example, cs_lyrics obtained $\leq$ 5 points from 20% of participants. According to the averaged scores of these specific 20% of participants, cs_lyrics were beaten by btl_wskr and by machine translations.

(b) In each tick x on the X-axis, we show the results from participants who gave **more than** x points to the cs_lyrics model. For example, cs_lyrics obtained $>$ 4 points from 96% of participants. The averaged scores of these 96% of participants say that cs_lyrics is the best-performing model with an 88% chance of being chosen in every inter-model comparison.

Figure 6.19: Filtered results by human-written song translations (cs_lyrics) supporters. The results are filtered by the number of times cs_lyrics was chosen by each respondent individually in the inter-model comparison. Besides the model results, we also include the thick grey line to show the percentage of participants rating the model the specific way the X-axis describes.

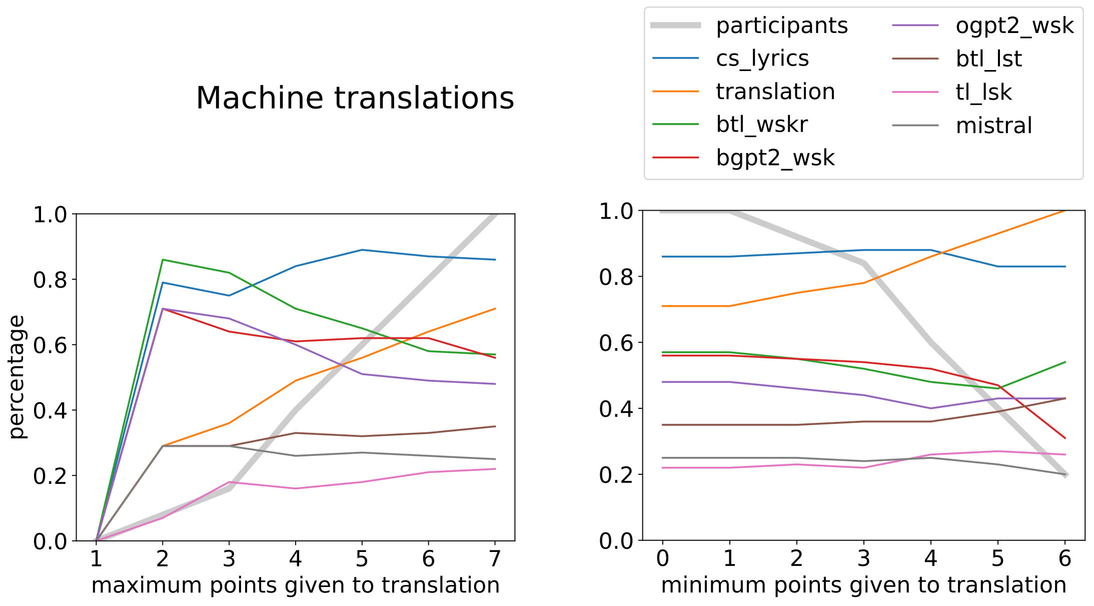


(a) In each tick x on the X-axis, we show the results from participants who gave **less than or equal to** x points to the btl_wskr model. For example, btl_wskr obtained $\leq$ 2 points from 12% of participants. According to the averaged scores of these specific 12% of participants, btl_wskr is the worst-performing model.

(b) In each tick x on the X-axis, we show the results from participants who gave **more than** x points to the btl_wskr model. For example, btl_wskr obtained $>$ 5 points from 12% of participants. The averaged scores of these 12% of participants say that btl_wskr is the best-performing model with a 90% chance of being chosen in each comparison.

Figure 6.20: Filtered results by btl_wskr supporters. The results are filtered by the number of times btl_wskr was chosen by each respondent individually in the inter-model comparison. Besides the model results, we also include the thick grey line to show the percentage of participants rating the model the specific way the X-axis describes.

number of people who don't like the model. The majority of participants (76%) gave the model between 3 to 5 points. We can see in Figure 6.20a that when the score of btl_wskr is low, the scores of other generative models are higher, meaning that in those cases, the btl_wskr model was probably unlucky. On the other hand, in Figure 6.20b we can see that when the score of the btl_wskr model goes up, the scores of other models generating whole song sections also rise and the score of the machine translator drops drastically. The averaged results of 68% of participants put btl_wskr above the machine translator.



(a) In each tick x on the X-axis, we show the results from participants who gave **less than or equal to** x points to the translation model. For example, translation obtained ≤ 3 points from 40% of participants. According to the averaged scores of these specific 40% of participants, the machine translations sit right between the btl_wskr model and the few-shotted mistral model.

(b) In each tick x on the X-axis, we show the results from participants who gave **more than** x points to the translation model. For example, translation obtained > 5 points from 40% of participants. The averaged scores of these 40% of participants say that translation is the best-performing model with a 93% chance of being chosen in every inter-model comparison.

Figure 6.21: Filtered results by machine translation supporters. The results are filtered by the number of times translation was chosen by each respondent individually in the inter-model comparison. Besides the model results, we also include the thick grey line to show the percentage of participants rating the model the specific way the X-axis describes.

Finally, we filter the results by the number of times machine translation was chosen as the better option in the inter-model comparison. As we can see in Figure 6.21, machine translation is very stable and none of the participants gave it 0 points. Participants who do not like the machine translation outputs prefer the outputs of models generating whole sections, however, the scores of the models generating sections line by line stay low (see Subfigure 6.21a). On the other hand, liking the machine translation outputs does not influence the average of participants' ratings of the generative language models (see Figure 6.21b).

source	points out of 7	chosen in % of cases
translation	**6.50**	**93**
cs_lyrics	5.80	83
bgpt2_wsk	3.30	47
btl_wskr	3.20	46
ogpt2_wsk	3.00	43
btl_lst	2.70	39
tl_lsk	1.90	27
mistral	1.60	23

Table 6.26: Averaged results of the manual evaluation from 10 participants who rated the translation model with more than 5 points.

source	points out of 7	chosen in % of cases
cs_lyrics	**6.20**	**89**
btl_wskr	**4.53**	**65**
bgpt2_wsk	**4.33**	**62**
translation	3.93	56
ogpt2_wsk	3.60	51
btl_lst	2.27	32
mistral	1.87	27
tl_lsk	1.27	18

Table 6.27: Averaged results of the manual evaluation from 15 participants who rated the translation model with less than or equal to 5 points.

20% of participants always chose a translator when they could (translator obtained 7 points). The averaged scores of 52% participants show that the machine translation is preferred to the human-translated song lyrics.

We observed that the participants can be split into two groups: one that prefers the machine translations and one that does not. We chose the criterion of this split to be whether each individual participant rated the machine translator with more than 5 points or not. The group of participants who rated translations with more than 5 points contains 10 people, while the other group, where the translator was rated with 5 or fewer points contains the other 15 people. We can see in Tables 6.26 and 6.27 how the averaged results of these two groups differ.

The group of translator supporters (10 out of 25 participants) likes the results of the translator more than the original human-translated Czech lyrics. They also do not differentiate much between the individual models generating whole sections at once. The other group (15 out of 25 participants) prefers the human-translated Czech lyrics, followed by the two bigger models fine-tuned on whole sections. There is a step in scores between these two, and the smaller ogpt2_wsk model. The translator is also chosen a decent amount of times in comparison with the other models.

This division of people into two groups of preferences could be due to the various knowledge of the original songs or different musical backgrounds. However, we think that the main reason is the different preferences of the criteria of singable translations. As we discussed in Section 1.1.3, singable translations should meet four criteria, all equally, none should be favoured and none should be neglected. These criteria are singability, rhyme, sense and naturalness. Each of the evaluated models meets the criteria differently and it is up to person's preference to choose which criteria are the most important to them. From our observation, the generative models better meet the criteria of rhyme and singability, while the machine translator better meets the criteria of sense and naturalness.

6.5 Discussion

In this section, we will discuss the combined results of the automatic and the manual evaluation.

Let us start with the 10-shotted Mistral model. While the semantic similarity was on par with the target, the rest of the criteria of singable translations (see Section 1.1.3) were not fulfilled according to the automatic evaluation. There was a big improvement in the syllable distance score after deploying the *stopwords* postprocessing function (see Table 6.15), which implies that the output sections were heavily stopwords-padded. Based on the results of human evaluation, the 10-shotted Mistral did not yield satisfactory results.

Next, let us look at the TL-LSK model. Based on the automatic metrics, this model had exceptional syllable counts (Tab. 6.16) and rhyme schemes (Tab. 6.18). The semantic similarity was average. Based on the results of the manual evaluation, the naturalness and overall singability were poor. Judging by the above-average automatic metrics results and below-average manual evaluation results, the model probably tried to fill in the structure without keeping any of the naturalness, focusing on some aspects of singable covers too much and on some too little. BTL-LSK had similar scores, except having semantic similarity better than the target (see Table 6.19). It did a bit better in the manual evaluation, but not by much.

The models generating whole song sections at once (BTL-WSKR, BGPT2-WSK and OGPT2-WSK) were slightly worse than the human-translated Czech lyrics when evaluated both by the automatic metrics and human evaluators. This shows that these models produce overall balanced results, that still do not reach the quality of human-made cover versions, but safely beat the random baseline. The order of the models based on the automatic evaluation corresponds to the order of the models based on the manual evaluation. Contrary to the models trained on individual lines, these models preserved some sense of naturalness on their own, keeping their good spot in both automatic and manual evaluation.

Lastly, according to the automatic metrics (Tab. 6.2), raw machine translations had great semantic similarity, neither ideal nor terrible syllable counts and a weak rhyme scheme. Nevertheless, the naturalness of the Lindat machine translation is excellent, so the overall score is not bad, just very one-sided. It showed during the manual evaluation when part of the participants preferred the natural-sounding and semantically accurate translation and part of people preferred the more poetic and structure-based adaptation.

7. Implementation and usage

In this chapter, we will describe the implementation and usage of both of our interfaces available for interacting with the cover-generating models. The first option is a command line interface, which supports running generation with the larger models and various settings. The second one is a web application that allows generating covers by the fine-tuned Czech-GPT2-Oscar model which is small enough to run locally. On top of that, it provides an interactive demo of the N-gram-based framework for lyrics generation. All code and data are in the attachments of this thesis, as well as in a GitHub repository[1], where we describe the structure of the codebase in a readme file. The state dictionary of the best fine-tuned Czech-GPT2-Oscar model (OGPT2-WSK) is also attached to the thesis as well as downloadable together with the best-performing CSTinyLlama model (BTL-WSKR) from Lindat's CLARIAH-CZ[2].

7.1 Command line interface

7.1.1 Implementation

The whole pipeline is implemented in Python. We are using the Huggingface Transformers library for large language models fine-tuning and inference through PyTorch. We are using RhymeTagger for rhyme detection, Lindat Translator for machine translation, and a range of other libraries and tools for smaller tasks.

7.1.2 Usage

In this section, we will describe the options of the command line interface. The program can be run by the following command:

```
python covermaker.py --arg1 value1 --arg2 value2 --arg3 value3 ...
```

The arguments are:

- `model`, default: `OSCAR_GPT2`, type: `string`
  - Name of the model. Options: OSCAR_GPT2, BUT_GPT2, TINYLLAMA, BUT_TINYLLAMA
- `model_path`, default: `./trained_models`, type: `string`
  - Path to the state dict of the fine-tuned model.
- `input_section`, default: `let it go,let it go ...`, type: `string`
  - Input section in English, lines divided by comma `‘,’` and sections divided by semicolon `‘;’`, e.g. "let it go,let it go,can't hold it back anymore,let it go,let it go,turn away and slam the door"

[1] https://github.com/stepankovab/Generation-of-Czech-Lyrics-to-Cover-Songs

[2] http://hdl.handle.net/11234/1-5507

- `prompt_type`, default: `5`, type: `int`
  - Prompt type the model was fine-tuned on. Submit a number. Options: syllables = 1, syllables ends = 2, keywords = 3, keywords ends = 4, syllables keywords = 5, syllables keywords ends = 6, syllables unrhymed = 7, syllables unrhymed ends = 8, syllables keywords rhymes = 13
- `from_dataset`, default: `False`, type: `bool`
  - Take test data from Bilingual human-translated lyrics dataset.
- `from_structures`, default: `False`, type: `bool`
  - Take test data from the pre-made song structures.
- `dataset_path`, default: `./../Data`, type: `string`
  - Path to the test data.
- `test_set_size`, default: `0`, type: `int`
  - Number of samples taken from the test dataset, 0 means all.
- `generation_method`, default: `whole`, type: `string`
  - The method of generating a section the model was finetuned on, or fewshot. Options: whole, lines, fewshot.
- `nshot`, default: `10`, type: `int`
  - Number of examples when using few-shot as generation method.
- `rhymer`, default: `1`, type: `string`
  - The rhyme detector to be used. Options: RhymeFinder = 1, RhymeTagger = 2, Same-word RhymeTagger = 3.
- `choose_best`, default: `10`, type: `int`
  - Choose best postprocessing technique - the number of generated outputs to choose the best from.
- `postprocess_stopwords`, default: `False`, type: `bool`
  - Posrprocess each output by trying to correct the length by removing/adding stopwords.
- `results_path`, default: `./results_dicts`, type: `string`
  - Path to folder to save the results when taking test data from dataset.

When the program is called without the default arguments (i.e. `python covermaker.py`), a cover of the first chorus of the song ‘Let it go’ from Frozen is generated by the OGPT2-WSK model with the postprocessing method of choosing the best out of 10 generated covers.

7.2 Web application

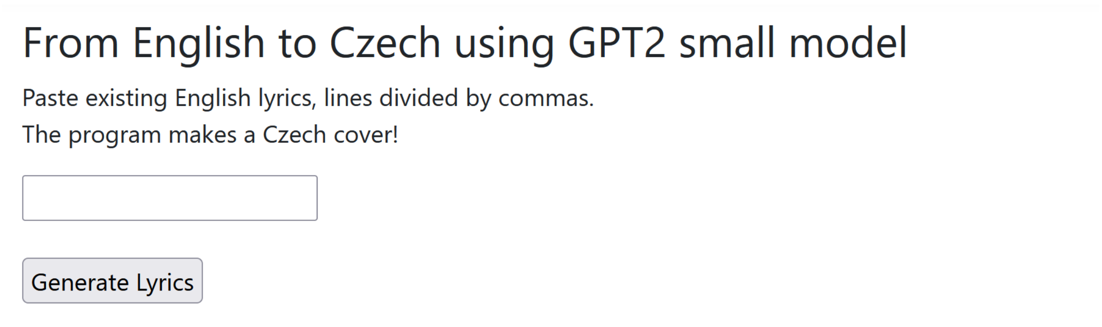
7.2.1 Implementation

As part of this work, we provide a web application based on the ASP.NET framework coded in Csharp. The application is used to visualize our work. It contains two GET endpoints, `http://localhost:5000/Generator/RewriteLyricsGPT2` for generating a Czech cover of an English song by a fine-tuned GPT2 small model and the `http://localhost:5000/Generator/CustomLyrics` for visualizing the N-gram generation approach. In the first part, the same Python scripts are being called as in the CLI application described in section 7.1. The second part is implemented purely as a component of the ASP.NET application.

7.2.2 Usage

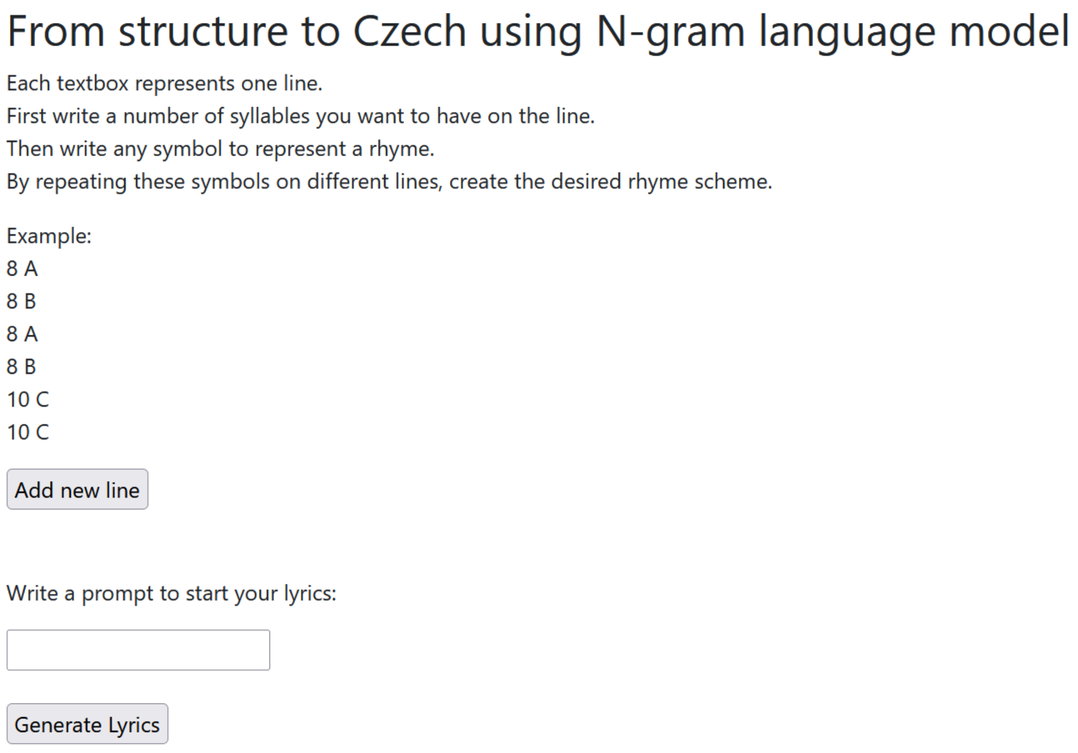
As stated above, the web application offers two kinds of models for cover song generation. To use the Ngram-based framework, nothing needs to be done. To use the GPT2-based framework, the path to Python, the path to the folder with the Python scripts and the path to the OGPT2-WSK model has to be set in the `config.json` file.

After starting the 'webapplication.exe', copy the generated URL into your favourite web browser. For generating lyrics by our GPT2 small model, fill in the text field in the 'From English to Czech using GPT2 small model' and click the 'Generate Lyrics' button (see Figure 7.1). To use the N-gram based framework, add lines to the structure by clicking the 'Add new line' button and describe the desired structure. Provide prompt for start of generation. Then, click on the 'Generate Lyrics' button (see Figure 7.2). An example of an output generated by the N-gram-based framework can be seen in Figure 7.3. There is also the option of generating a specific line again, as shown in Figure 7.4.


From English to Czech using GPT2 small model

Paste existing English lyrics, lines divided by commas.
The program makes a Czech cover!

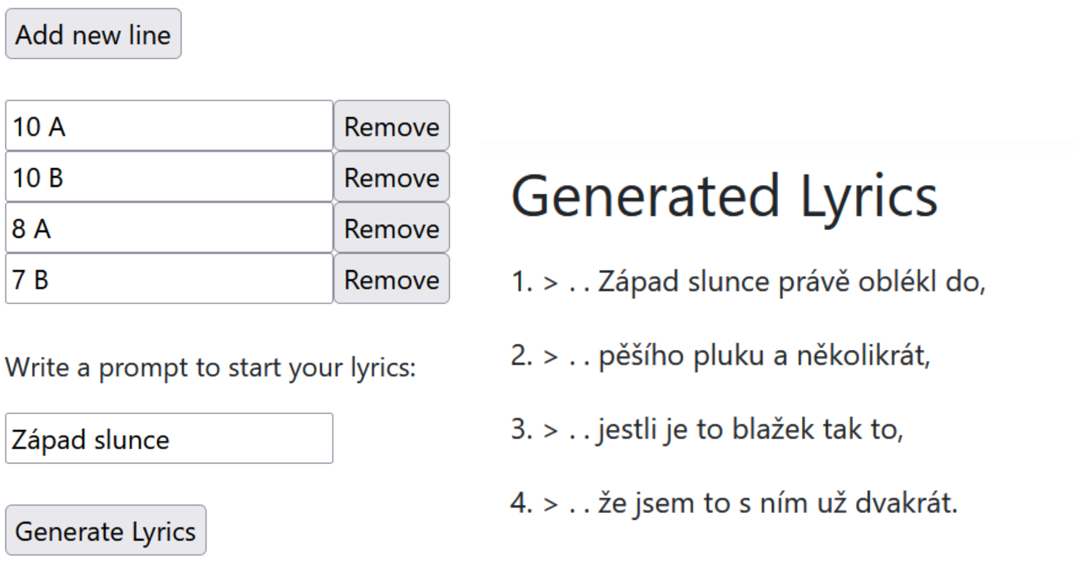
Figure 7.1: Web application. To generate a Czech cover of an English song using the OGPT2-WSK, write the English lyrics into the textbox, lines divided by commas (`','`)


From structure to Czech using N-gram language model

Each textbox represents one line.
First write a number of syllables you want to have on the line.
Then write any symbol to represent a rhyme.
By repeating these symbols on different lines, create the desired rhyme scheme.

Example:
8 A
8 B
8 A
8 B
10 C
10 C

Figure 7.2: Web application. To generate Czech lyrics to a given structure by the framework using the N-gram language model, add lines to the structure by clicking the ‘Add new line’ button and describe the desired structure. Provide prompt for start of generation.



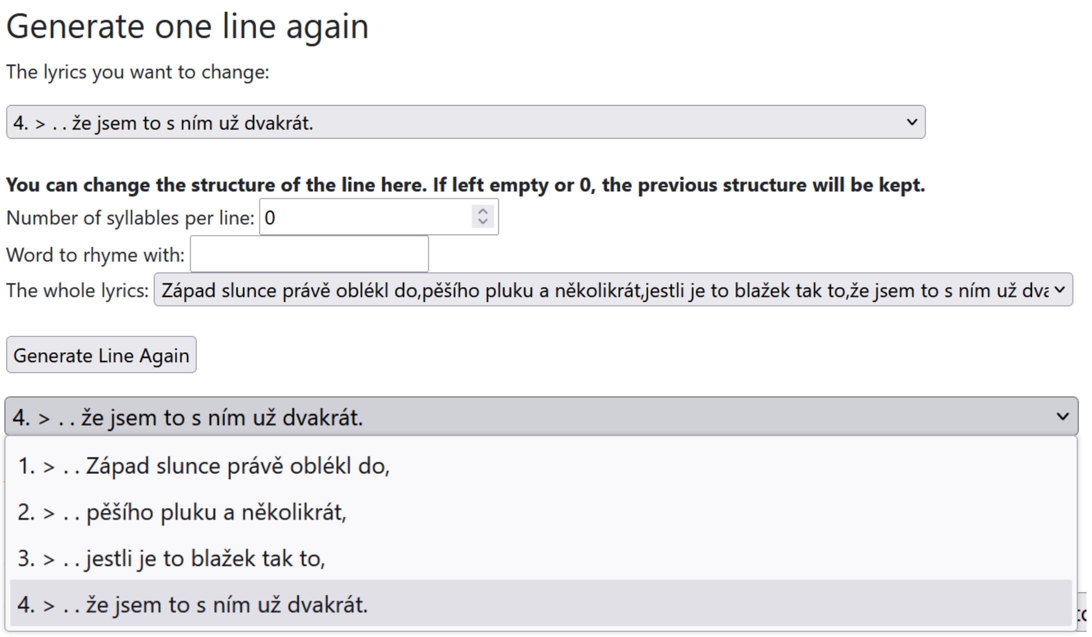
10 A
10 B
8 A
7 B

Západ slunce

Generated Lyrics

1. > . . Západ slunce právě oblékl do,
2. > . . pěšího pluku a několikrát,
3. > . . jestli je to blažek tak to,
4. > . . že jsem to s ním už dvakrát.

Figure 7.3: An example of a described structure and a generated output.


Generate one line again

The lyrics you want to change:

4. > . . že jsem to s ním už dvakrát.

You can change the structure of the line here. If left empty or 0, the previous structure will be kept.

Number of syllables per line: 0

Word to rhyme with:

The whole lyrics: Západ slunce právě oblékl do,pěšího pluku a několikrát,jestli je to blažek tak to,že jsem to s ním už dva

Generate Line Again

4. > . . že jsem to s ním už dvakrát.

1. > . . Západ slunce právě oblékl do,

2. > . . pěšího pluku a několikrát,

3. > . . jestli je to blažek tak to,

4. > . . že jsem to s ním už dvakrát.

Figure 7.4: In case the model generates an inadequate line, there is the option of generating only that line again, while also providing the option to adjust the number of syllables, or a word the line should rhyme with.

Conclusion

In this work, we aimed to explore the automatic generation of Czech lyrics following the same melody as the English original of the song. In other words, we aimed to create Czech cover versions.

First, we thoroughly researched the problem. Then we created two datasets, one consisting of Czech and English paired and aligned lyrics, and the second one consisting purely of Czech song lyrics. The data for both datasets were parsed, filtered and automatically annotated (adding information such as syllable counts, rhyme scheme and keywords).

We experimented with generative decoder-based models, trying different approaches to leveraging the outputs. The models were fine-tuned on the song sections with various information provided via prompting. We also tried few-shotting techniques. At inference, we extracted the structure from the source English lyrics and used it to prompt the language model to generate the Czech cover. In the end, we evaluated and compared the performances of the proposed models with the random baseline, machine translations and the official Czech adaptations of the English songs. These were evaluated by metrics, both adapted from different publications and our own. The best-performing models were also evaluated manually by human evaluators. Additionally, we attached a web application that visualizes the results of our work.

Our experiments have consistently shown that fine-tuning smaller generative language models leads to superior results compared to using few-shot learning on a larger generative language model. While acknowledging the potential for improved performance with fine-tuning larger models, it is clear that fine-tuning smaller models is advantageous given our resources, as these smaller but specialized models are more usable than the general larger ones.

Language models can learn to rhyme on their own. When outputs of a certain length are desired, the model has to get explicit syllable counts on the input. The same applies when we want the outputs to have a specific meaning. In that case, providing keywords significantly improves the performance. In comparison, the results of the models with, and without the provided rhyme scheme both yield comparable results. This shows that models can learn to rhyme based on observations of rhyming texts only, without any explicit reinforcement.

Fine-tuning language models on whole song sections yields overall better results than fine-tuning the models to generate sections line by line. Fine-tuning a language model to generate a song section line by line supports the rhyme of the lyrics, but negatively impacts the sense and naturalness of the lyrics, as the models can not learn from the wider context. Generating whole lyric sections lets the model learn the structure without forgetting the underlying naturalness of the language, both in the flow of the ideas throughout the section or for example in the aforementioned rhyme scheme adherence. The manual evaluation results show that naturalness plays a crucial role when assessing the overall quality of the lyrics, therefore putting the more naturally sounding models (generating whole sections) way above the less naturally sounding models (generating sections line by line). For some people, naturalness is so important that the naturalness of a machine translation even outweighs the singability of song lyrics.

Our research reveals the subjectivity in individual preferences when evaluating covers. Human evaluators can be divided into two groups: one favours raw machine translation, while the other favours our fine-tuned language model's outputs and human-translated song lyrics. The question arises: is it even possible to accurately and objectively measure the quality of a cover? While for example, the musical film industry prefers the highly singable song adaptation due to having to lip-sync with the pre-recorded material, others could prefer more of a translation approach underlining the meaning and naturalness of a language. It is unclear whether the ideal set of metrics for measuring the quality of song covers even exists.

Finally, we can conclude that while human-translated song lyrics are singable and widely used in pop culture, they are not translations but merely adaptations. This means that there is a lot of room for improvement not only to get on the level of human-translated lyrics but also to surpass them.

Future work

We demonstrated the possibility of generating Czech covers of English songs through the extraction of relevant lyric characteristics and fine-tuning of generative large language models. Our method has shown promising performance compared to baseline approaches. However, it is important to acknowledge that our generated covers still fall short of the quality achieved by human-written covers. Despite this gap, our approach represents a significant step forward in the development of automated cover generation methods.

For future work, our results indicate that although this is not a traditional translation task, we should not overlook the potential of Machine Translation. Therefore, it is essential to explore the performance of machine translation models specifically trained for this task. Additionally, investigating the effectiveness of larger generative models would be valuable. An intriguing avenue for research would be to combine these two approaches, integrating the injection of meaning vectors into the large language models instead of only providing keywords. Furthermore, inventing a reliable metric for measuring the naturalness of generated lyrics would be beneficial.

Bibliography

Alejandro Benito-Santos, Adrián Ghajari, Pedro Hernández, Víctor Fresno, Salvador Ros, and Elena González-Blanco. Lyricsim: A novel dataset and benchmark for similarity detection in spanish song lyrics. *arXiv preprint arXiv:2306.01325*, 2023.

L. Dean Bye. *Mel Bay Presents Student's Musical Dictionary.* Open Book Publishers, 1993. ISBN: ISBN 0-87166-313-9.

Lukáš Chaloupský. Automatic generation of medical reports from chest x-rays in czech. *Charles University, Faculty of Mathematics and Physics*, 2022.

Don Cusic. In defense of cover songs. *Popular Music and Society*, 28(2):171–177, 2005. doi: 10.1080/03007760500045279. URL `https://doi.org/10.1080/03007760500045279`.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Belén Díaz-Agudo, Pablo Gervás, and Pedro A González-Calero. Poetry generation in colibri. In *Advances in Case-Based Reasoning: 6th European Conference, ECCBR 2002 Aberdeen, Scotland, UK, September 4–7, 2002 Proceedings 6*, pages 73–87. Springer, 2002.

Martin Fajčík, Martin Dočekal, Jan Doležal, Karel Beneš, and Michal Hradiš. Benczechmark: Machine language understanding benchmark for czech language, 2024. URL `https://huggingface.co/BUT-FIT`.

Michael G. Farringdon. Symposium on the uses of the computer in literary research: A conference report. *Computers and the Humanities*, 4(5):315–317, 1970. ISSN 00104817. URL `http://www.jstor.org/stable/30199382`.

Edward Flemming. The phonetics of schwa vowels. *Phonological weakness in English*, 493:78–95, 2009.

Johan Franzon. Choices in song translation: Singability in print, subtitles and sung performance. *The Translator*, 14(2):373–399, 2008.

Dmitriy Genzel, Jakob Uszkoreit, and Franz Och. "poetic" statistical machine translation: Rhyme and meter. In Hang Li and Lluís Màrquez, editors, *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 158–166, Cambridge, MA, October 2010. Association for Computational Linguistics. URL `https://aclanthology.org/D10-1016`.

Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. Rwc music database: Popular, classical and jazz music databases. In *Ismir*, volume 2, pages 287–288, 2002.

Fenfei Guo, Chen Zhang, Zhirui Zhang, Qixin He, Kejun Zhang, Jun Xie, and Jordan Boyd-Graber. Automatic song translation for tonal languages. *arXiv preprint arXiv:2203.13420*, 2022. URL `https://aclanthology.org/2022.findings-acl.60.pdf`.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

Jack Hopkins and Douwe Kiela. Automatically generating rhythmic verse with neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 168–178, 2017.

Dan Jurafsky and James H Martin. Speech and language processing. 3rd, 2024. URL `https://web.stanford.edu/~jurafsky/slp3/`.

Muhammad Qasim Khan, Abdul Shahid, M Irfan Uddin, Muhammad Roman, Abdullah Alharbi, Wael Alosaimi, Jameel Almalki, and Saeed M Alshahrani. Impact analysis of keyword extraction using contextual word embedding. *PeerJ Computer Science*, 8:e967, 2022. URL `https://peerj.com/articles/cs-967/`.

Haven Kim, Jongmin Jung, Dasaem Jeong, and Juhan Nam. K-pop lyric translation: Dataset, analysis, and neural-modelling. *arXiv preprint arXiv:2309.11093*, 2023a.

Haven Kim, Kento Watanabe, Masataka Goto, and Juhan Nam. A computational evaluation framework for singable lyric translation. *arXiv preprint arXiv:2308.13715*, 2023b. URL `https://arxiv.org/pdf/2308.13715.pdf`.

Hsin-Pei Lee, Jhih-Sheng Fang, and Wei-Yun Ma. iComposer: An automatic songwriting system for Chinese popular music. In Waleed Ammar, Annie Louis, and Nasrin Mostafazadeh, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 84–88, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-4015. URL `https://aclanthology.org/N19-4015`.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*, 2015. URL `https://arxiv.org/pdf/1510.03055.pdf`.

Piji Li, Haisong Zhang, Xiaojiang Liu, and Shuming Shi. Songnet: Rigid formats controlled text generation. *arXiv preprint arXiv:2004.08022*, 2020.

Kai-Ling Lo, Rami Ariss, and Philipp Kurz. Gpoet-2: A gpt-2 based poem generator. *arXiv preprint arXiv:2205.08847*, 2022.

Peter Low. Singable translations of songs. *Perspectives*, 11(2):87–103, 2003. doi: 10.1080/0907676X.2003.9961466. URL `https://doi.org/10.1080/0907676X.2003.9961466`.

Dominik Macháček. Sekáček. `https://github.com/Gldkslfmsd/sekacek`, 2014.

P. D. Magnus. *A Philosophy of Cover Songs.* Open Book Publishers, 2022. URL `https://doi.org/10.11647/OBP.0293`. ISBN: 978-1-80064-424-3.

Eric Malmi, Pyry Takala, Hannu Toivonen, Tapani Raiko, and Aristides Gionis. Dopelearning: A computational approach to rap lyrics generation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 195–204, 2016.

Hisar Manurung. An evolutionary algorithm approach to poetry generation, 2004.

Longshen Ou, Xichu Ma, Min-Yen Kan, and Ye Wang. Songs across borders: Singable and controllable neural lyric translation. *arXiv preprint arXiv:2305.16816*, 2023. URL `https://arxiv.org/pdf/2305.16816.pdf`.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002. URL `https://aclanthology.org/P02-1040.pdf`.

Petr Plecháč. A collocation-driven method of discovering rhymes (in czech, english, and french poetry). *Taming the Corpus: From Inflection and Lexis to Interpretation*, pages 79–95, 2018. URL `https://link.springer.com/chapter/10.1007/978-3-319-98017-1_5`.

Martin Popel, Marketa Tomkova, Jakub Tomek, Łukasz Kaiser, Jakob Uszkoreit, Ondřej Bojar, and Zdeněk Žabokrtskỳ. Transforming machine translation: a deep learning system reaches news translation quality comparable to human professionals. *Nature communications*, 11(1):1–15, 2020. URL `https://www.nature.com/articles/s41467-020-18073-9`.

Maja Popović. chrf: character n-gram f-score for automatic mt evaluation. In *Proceedings of the tenth workshop on statistical machine translation*, pages 392–395, 2015. URL `https://aclanthology.org/W15-3049.pdf`.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

Ashish Vaswani, 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.

Kento Watanabe, Yuichiroh Matsubayashi, Satoru Fukayama, Masataka Goto, Kentaro Inui, and Tomoyasu Nakano. A melody-conditioned lyrics language model. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 163–172, 2018. URL `https://aclanthology.org/N18-1015.pdf`.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. Tinyllama: An open-source small language model, 2024.

Xu Zou, Da Yin, Qingyang Zhong, Hongxia Yang, Zhilin Yang, and Jie Tang. Controllable generation from pre-trained language models via inverse prompting. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2450–2460, 2021.

Jitka Štindlová. Dělení slov v češtině pomocí strojů. *Naše řeč*, pages 23–32, 1968. URL `http://nase-rec.ujc.cas.cz/archiv.php?art=5348`.

List of Figures

1.1 The original transformer model architecture from the paper *Attention is all you need* by Vaswani et al. [2017] 9

4.1 English and Czech version of “Do you hear the people sing?” from Les Miserables with unalignable lines 21

5.1 section from *Surface Pressure* from the *Encanto* Musical, written by Lin-Manuel Miranda and its Czech translation 27
5.2 Two examples of lyrics generated by the N-gram-based model with a given structure of 8 syllables per line and rhyme scheme of ABAB 29

6.1 Syllable distance of the random baseline and the target 39
6.2 syllable accuracy of the random baseline and the target 39
6.3 Rhyme scheme agreement of the random baseline and the target . 40
6.4 Rhyme scheme accuracy of the random baseline and the target . . 40
6.5 Phoneme repetition difference of the random baseline and the target 40
6.6 Semantic similarity of the random baseline and the target 41
6.7 Keyword similarity of the random baseline and the target 41
6.8 Line-by-line keyword similarity of the random baseline and the target 41
6.9 BLEU2 score of the random baseline and the target 42
6.10 chrF score of the random baseline and the target 42
6.11 The results of individual metrics from evaluation function taken after each epoch of fine-tuning 46
6.12 Generated Czech covers of the English original song sections by the BTL-WSKR model with *choose best* of 10 and *stopwords* post-processing. 56
6.13 Generated Czech cover of the English original song section by the BGPT2-WSK model with *choose best* of 10 and *stopwords* post-processing. 56
6.14 Generated Czech cover of the English original song section by the OGPT2-WSK model with *choose best* of 10 and *stopwords* post-processing. 57
6.15 Generated Czech cover of the English original song section by the TL-LSK model with *choose best* of 10 and *stopwords* postprocessing. 57
6.16 Generated Czech cover of the English original song section by the BTL-LST model with *choose best* of 10 and *stopwords* postprocessing. 57
6.17 Generated Czech cover of the English original song section by the Mistral model with 10-shot prompting, *choose best* of 10 and *stopwords* postprocessing. 58
6.18 An example of comparison of BTL-WSKR output section and BGPT2-WSK output section, with the prompt ‘Lepsi:’ expecting the id of the better song section. 59

6.19 Filtered results by human-written song translations (cs_lyrics) supporters. The results are filtered by the number of times cs_lyrics was chosen by each respondent individually in the inter-model comparison. Besides the model results, we also include the thick grey line to show the percentage of participants rating the model the specific way the X-axis describes. 61
6.20 Filtered results by btl_wskr supporters. The results are filtered by the number of times btl_wskr was chosen by each respondent individually in the inter-model comparison. Besides the model results, we also include the thick grey line to show the percentage of participants rating the model the specific way the X-axis describes. 62
6.21 Filtered results by machine translation supporters. The results are filtered by the number of times translation was chosen by each respondent individually in the inter-model comparison. Besides the model results, we also include the thick grey line to show the percentage of participants rating the model the specific way the X-axis describes. 63

7.1 Web application. To generate a Czech cover of an English song using the OGPT2-WSK, write the English lyrics into the textbox, lines divided by commas (`‘,’`) 68
7.2 Web application. To generate Czech lyrics to a given structure by the framework using the N-gram language model, add lines to the structure by clicking the ‘Add new line’ button and describe the desired structure. Provide prompt for start of generation. 69
7.3 An example of a described structure and a generated output. . . . 69
7.4 In case the model generates an inadequate line, there is the option of generating only that line again, while also providing the option to adjust the number of syllables, or a word the line should rhyme with. 70

List of Tables

1.1 Types of rhymes commonly appearing in song lyrics 7
1.2 Example of 3-shot prompting a model to output a sentiment of a sentence in one word 11

5.1 Mapping of endings to a rhyme key for Czech RhymeFinder . . . 28
5.2 Training examples for fine-tuning LLM's to generate whole song sections with different information available 31
5.3 Training examples for fine-tuning LLMs to generate song sections line by line with different information available 33

6.1 Random baseline values and Target values of metrics from the evaluation function, measured on the Bilingual Song Lyrics dataset 38
6.2 Machine translator baseline values 43
6.3 Abbreviations used in Sections 6.3 and 6.4 44
6.4 Base models outputs, prompted by "Zapadající slunce" 45
6.5 Syllable distance on raw outputs 47
6.6 Syllable accuracy on raw outputs 47
6.7 Rhyme scheme agreement on raw outputs 48
6.8 Rhyme scheme accuracy on raw outputs 48
6.9 Semantic similarity on raw outputs 49
6.10 Keyword similarity on raw outputs 49
6.11 Line-by-line keyword similarity on raw outputs 49
6.12 Phoneme repetition distance on raw outputs 50
6.13 BLEU-2 on raw outputs 50
6.14 chrF on raw outputs 50
6.15 Syllable distance after postprocessing 52
6.16 Syllable accuracy after postprocessing 52
6.17 Rhyme scheme agreement after postprocessing 53
6.18 Rhyme scheme accuracy after postprocessing 53
6.19 Semantic similarity after postprocessing 53
6.20 Keyword similarity after postprocessing 54
6.21 Line-by-line keyword similarity after postprocessing 54
6.22 Phoneme repetition distance after postprocessing 55
6.23 BLEU2 after postprocessing 55
6.24 chrF after postprocessing 55
6.25 Averaged results of the manual evaluation from all 25 participants. 60
6.26 Averaged results of the manual evaluation from 10 participants who rated the translation model with more than 5 points. 64
6.27 Averaged results of the manual evaluation from 15 participants who rated the translation model with less than or equal to 5 points. 64