



**Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie**

**Wydział Informatyki**

**PRACA DYPLOMOWA**

**Generacja muzyki przy pomocy dużych modeli językowych**

**Music generation with Large Language Models**

<b>Autor:</b>	<b>Filip Ręka</b>
<b>Kierunek:</b>	<b>Informatyka — Data Science</b>
<b>Opiekun pracy:</b>	<b>dr hab. Maciej Smółka prof. AGH</b>

**Kraków, 2024**



*Tutaj możesz umieścić treść podziękowań. Tutaj możesz umieścić treść podziękowań. Tutaj możesz umieścić treść podziękowań. Tutaj możesz umieścić treść podziękowań.*



## Streszczenie

Duże modele językowe (ang. *Large Language Models* **LLM**) charakteryzują się zdolnością do generacji języka oraz innych zadań w przetwarzania języka naturalnego, takich jak na przykład klasyfikacja. Zdolność tą nabierają podczas czasochłonnego oraz intensywnego obliczeniowego treningu metodami samo odraz pół-nadzorowanego, podczas którego uczą się one relacji z wielkiej ilości dokumentów tekstowych. LLMy mogą zostać wykorzystane do generacji tekstu, formy generatywnej sztucznej inteligencji, poprzez pobieranie tekstu wejściowego i wielokrotne przewidywanie kolejnego tokenu lub słowa w tekście. Strukturę muzyki można porównać struktury tekstu pisanego, gdzie każda nuta odpowiada literze lub słowu, akordy zdaniom a dłuższe i sekwencje paragrafom. Poniższa praca, zamierza zbadać możliwości generacyjne LLMów wytrenowanych na muzycznych zbiorach danych.



## **Abstract**

Abstract in English [\[1\]](#) ...

# Spis treści

<b>Lista kodów źródłowych</b>	<b>xiii</b>
<b>1 Wstęp</b>	<b>1</b>
1.1 Motywacja . . . . .	1
1.2 Cel pracy . . . . .	1
1.3 Zarys pracy dyplomowej . . . . .	2
<b>2 Opis aktualnego stanu wiedzy</b>	<b>3</b>
2.1 Cyfrowa reprezentacja muzyki . . . . .	3
2.1.1 WAV i MP3 . . . . .	3
2.1.2 MIDI (ang. <i>Musical Instrument Digital Interface</i> ) . . . . .	5
2.1.3 Notacja ABC . . . . .	6
2.1.3.1 Tokenizacja plików MIDI . . . . .	7
2.1.4 Porównanie zapisu muzycznego . . . . .	8
2.2 Zbiory danych . . . . .	9
2.2.1 Johann Sebastian Bach Chorales . . . . .	9
2.2.2 The MAESTRO v3.0 . . . . .	9
2.2.3 Million Song Dataset . . . . .	9
2.3 STOA . . . . .	9
2.4 Architektury transformera . . . . .	10
2.4.1 Algorytm uwagi (ang. <i>attention</i> ) . . . . .	10
2.4.2 Warianty mechanizmu uwagi . . . . .	10
2.4.2.1 Self attention . . . . .	10
2.4.2.2 Multi-headed attention . . . . .	10
2.4.2.3 Flash attention . . . . .	10
2.4.3 Budowa transformera . . . . .	10
2.4.4 Modele tranformerowe . . . . .	10
2.4.4.1 <i>Classic</i> transformer . . . . .	10
2.4.4.2 SeqGAN . . . . .	10
2.4.4.3 Mistral . . . . .	10
2.5 Architektura <i>state space</i> . . . . .	10



2.5.1	Mamba . . . . .	10
2.5.2	Tutaj się rozdrobnić trzeba . . . . .	10
<b>3</b>	<b>Propozycja rozwiązania</b>	<b>13</b>
<b>4</b>	<b>Przeprowadzenie eksperymentów</b>	<b>15</b>
4.1	Opis <i>pipeline-u</i> . . . . .	15
4.2	Porównanie architektur użytych modeli . . . . .	15
4.3	Prezentacja otrzymanych wyników . . . . .	15
4.4	Porównanie wyników . . . . .	15
<b>5</b>	<b>Zakończenie</b>	<b>17</b>
<b>Dodatek A.</b>	<b>Typowe elementy składowe pracy dyplomowej z informatyki</b>	<b>19</b>
A.1	Tabele . . . . .	19
A.2	Rysunki . . . . .	21
A.2.1	Wewnętrzne . . . . .	21
A.2.2	Zewnętrzne . . . . .	22
A.3	Kody źródłowe . . . . .	22
A.4	Algorytmy . . . . .	24
A.5	Wzory . . . . .	24
A.5.1	Przykłady . . . . .	25
A.6	Twierdzenia i podobne struktury . . . . .	25
	<b>Uwagi Autora</b>	<b>27</b>
	<b>Bibliografia</b>	<b>29</b>

Zawartość spisu treści — tytuły rozdziałów oraz ich liczba zależą od tematyki pracy — należy ustalić z opiekunem pracy.

## Spis rysunków

2.1	Fragment chorału J.S. Bacha . . . . .	4
-----	---------------------------------------	---

2.2	Plik <i>.wav</i> otworzony w programie Audacity. . . . .	4
2.3	Muzyka zapisana w pliku MIDI . . . . .	5
2.4	Zapis wielu głosów w notacji ABC. . . . .	6
2.5	Prosta sekwencja muzyczna . . . . .	8
2.6	Reprezentacja muzyki w postaci tokenów . . . . .	8
2.7	Schemat transformera. . . . .	11
2.8	Schemat modelu Mamba. . . . .	12
A.1	Prosty rysunek <i>TikZ</i> . . . . .	21
A.2	Bardziej złożony rysunek <i>TikZ</i> . . . . .	21
A.3	Logo Wydziału Informatyki. . . . .	22

## Spis tabel

2.1	Porównanie różnych zapisów muzyki . . . . .	8
A.1	Pomiary zużycia energii elektrycznej. . . . .	19
A.2	Tabela, która zawiera dużą liczbę wierszy. . . . .	19
A.3	Tabela zawierająca długi tekst. . . . .	20

# Lista algorytmów

1    Disjoint decomposition. . . . . 24



# Lista kodów źródłowych

A.1	Przykładowy kod źródłowy sformatowany za pomocą pakietu 'listings' . . .	23
A.1.	Przykładowy listing sformatowany za pomocą pakietu 'minted' . . . . .	23



# 1. Wstęp

Muzyka od zawsze stanowiła istotny element ludzkiego życia, inspirując, emocjonując i łącząc ludzi na różnych poziomach. Tradycyjnie proces tworzenia muzyki był zarezerwowany dla utalentowanych muzyków i kompozytorów, którzy posiadali wyjątkowy dar tworzenia dźwięków i melodii. Jednakże, w erze cyfrowej i rozwoju sztucznej inteligencji, pojawiają się nowe możliwości w dziedzinie generacji muzyki. Modelowanie generatywne, będące obszarem sztucznej inteligencji, pozwala na tworzenie nowych danych, w tym również muzyki, na podstawie wzorców i reguł wykrytych w zbiorze treningowym. Dynamiczny rozwój dziedziny przetwarzania języka naturalnego (*NLP*) ukazuje skuteczność tego podejścia w problemach generacji danych sekwencyjnych, do których należy właśnie tekst oraz muzyka.

## 1.1. Motywacja

Przez ostatnie lata widzimy szybki rozwój dużych modeli językowych (*LLM*), które w szczególności dobry sposób radzą sobie z rozumieniem tekstu w wielu językach. Modele te uczone są na wielkich korpusach danych, przez co są w stanie nauczyć się słownictwa oraz zasad gramatyki, a następnie na podstawie *promptu* udzielonego przez użytkownika wygenerować odpowiednią odpowiedź. Modele te znalazły zastosowanie w maszynowej translacji tekstu, analizie sentymentu, odpowiadaniu na pytania użytkownika czy nawet generacji kodu na podstawie poleceń oraz kontekstu nawet dużych projektów. Muzyka swoją strukturą jest bardzo podobna do tekstu. Każda nuta, tak jak słowo, nie tylko jest zależna od tych, które występują przed nią, ale również od szerszego kontekstu utworu jak również i tonacji w której dany utwór został napisany. Wykonując to porównanie, można łatwo zauważyć, dlaczego modele analizujące tekst oraz języki pisane są potencjalnie dobrymi kandydatami do próby analizy i generacji muzyki.

## 1.2. Cel pracy

Celem pracy jest przyjrzenie się dużym modelom językowym oraz ich zastosowaniu w celu generacji muzyki. Zawierać będzie opis modeli, które obecnie są wykorzystywane w architekturach *LLM*-ów oraz analizę odpowiadającą na pytanie dlaczego dane rozwiązanie sprawdza się w przedstawionym w pracy problemie. Następnie zostanie przedstawiony problem analizy

oraz generacji muzyki oraz jej możliwe cyfrowe reprezentacje. Zostaną również przedstawione dostępne zbiory danych, na podstawie których modele mogą zostać trenowane. Praca przedstawi wyniki użytych metod w celu ustalenia, który model i która reprezentacja muzyki najlepiej nadaje się do problemu przedstawionego w pracy.

### 1.3. Zarys pracy dyplomowej

**Uwaga 1.1.** Tutaj napiszę dopiero jak będzie więcej treści bo chyba troszkę bez sensu na razie pisać o strukturze jak jej jeszcze nie ma "\\_(ツ)\_/"



## 2. Opis aktualnego stanu wiedzy

### 2.1. Cyfrowa reprezentacja muzyki

W kulturze zachodniej nuty zapisuje się na pięcioliniach ułożonych jedna pod drugą. Jeśli muzyka jest wielogłosowa typowo pięciolinie są okalane przy pomocy nawiasów klamrowych. Na pięciolinia umieszczone są takie elementy jak nuty, pauzy oznaczenia dynamiki, tempa i inne znaki dotyczące oznaczenia w jaki sposób należy wykonać dany utwór. Najpierw muzykę zapisywano na papierze, jednak po pojawieniu się komputerów zapis nutowy jak i samo nagranie odtworzenia mogło zostać zapisane na dysku. Nuty typowo są zapisywane jako obrazy lub pliki PDF, jednak dane zapisane w taki sposób nie nadają się do edycji lub przetwarzania przez algorytmy komputerowe. Jednym z pierwszych ustandaryzowanych formatów był *MusicXML*, który przy pomocy zwykłych tagów XML opisywał notację muzyczną.

Przykładowy zapis muzyki zapisano na obrazku 2.1. Przedstawiony fragment składa się z czterech głosów grających na raz oraz z czterech taktów.

#### 2.1.1. WAV i MP3

WAV (ang. *Waveform audio format*) jest binarnym zapisem plików, który jest znany z tego, że jest w stanie zapisywać dźwięk nie używając żadnego algorytmu kompresji. W związku z tym rozmiary tych plików są bardzo duże, co sprawia, że ich przechowywanie wymaga znacznego miejsca na dysku co może być problematyczne, kiedy zbiór danych jest bardzo duży. Plik WAV jest najrzetelniejszą cyfrową reprezentacją dźwięku analogowego. Tak duży rozmiar danych zawdzięcza się temu, że dźwięk jest zapisywany z częstotliwością 44.1 kHz, czyli 44100 sampli na sekundę. Format został stworzony w roku 1991 roku, jest jednym z najbardziej rozpowszechnionych formatów i jest obsługiwany przez praktycznie każde oprogramowanie edycji dźwięku. W celu wizualizacji dokonano syntezy?? utworu przedstawionego w postaci nutowej na obrazku 2.1 w związku z czym powstała fala dźwiękowa przedstawiona na grafice 2.2.

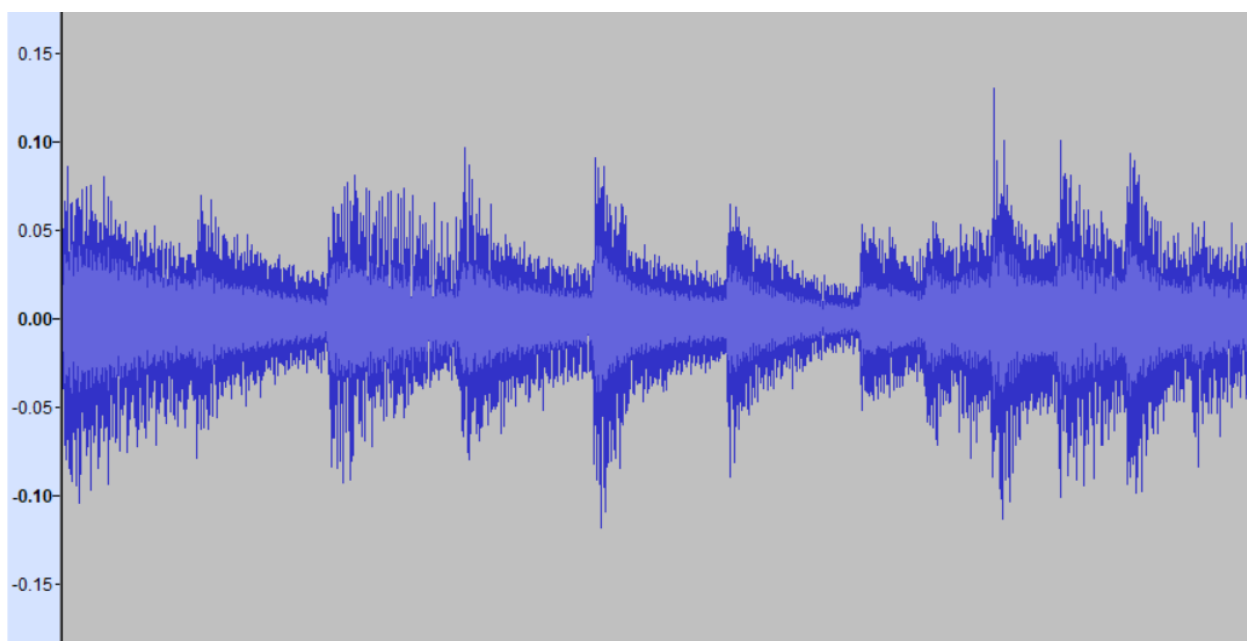
Często nie jest potrzebne przechowywanie bezstratne dźwięku, ponieważ i tak większość informacji, można zachować przy użyciu mniejszej ilości danych. Jednym z najpopularniejszych standardów kodowania stratnego jest MP3. Kompresja zmniejsza dokładność kodowania jak i również “ucina” częstotliwości, które nie są słyszalne dla człowieka. Stratne kodowanie próbuje zachować równowagę pomiędzy jakością dźwięku a rozmiarem pliku. W

## Chorał no. 7

J.S. Bach



Rysunek 2.1.: Fragment chorału J.S. Bacha



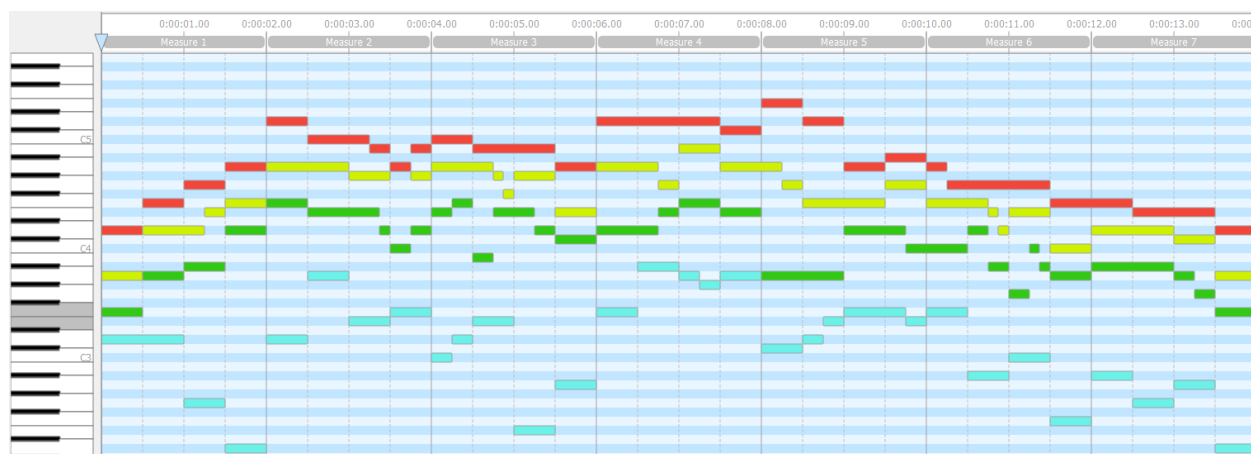
Rysunek 2.2.: Plik .wav otworzony w programie Audacity.

kontekście uczenie maszynowego, zmniejszona ilość sampli przy zachowaniu większości informacji jest zjawiskiem pożądanym, ponieważ udaje nam się przynajmniej częściowo rozwiązać problem związany z “przekleństwem wymiarowości”.

### 2.1.2. MIDI (ang. *Musical Instrument Digital Interface*)

MIDI jest standardem, który opisuje protokół komunikacji, interfejs cyfrowy oraz złącze pozwalające połączyć ze sobą elektroniczne instrumenty, komputery oraz inne muzyczne peryferia. Pojedynczy kabel MIDI jest w stanie przekazać informacje na temat szesnastu na raz nadających kanałów z czego każdy może pochodzić z innego instrumentu. Każda interakcja z instrumentem, czyli przykładowo naciśnięty klawisz, szarpnięta struna, jest zapisana jako “*event*”, który zachowuje wartości takie jak konkretny znacznik czasu, wysokość dźwięku czy jego głośność. Dane pochodzące z urządzeń są zapisywane w specjalnym pliku, głównie o rozszerzeniu *.mid* lub *.midi*. Plik pozwala na przechowanie, rozpowszechnianie jak i również edycje dźwięków. W związku z tym, że w pliku nie jest przechowywany zapis konkretnej fali dźwiękowej zapisanej przez mikrofon, pozwala to na późniejszą zmianę np. instrumentu, który będzie odtwarzał zapisane dźwięki.

Typowym przedstawieniem zapisanych danych jest tak zwany *piano roll*, który można porównać do dwuwymiarowego układu współrzędnych, gdzie współrzędna horyzontalna opisuje czas, a horyzontalna konkretne dźwięki pokazane jako klawisze pianina. Przykładowe porównanie zapisu MIDI z zapisem nutowym przedstawiono na grafice 2.3 oraz 2.1.



Rysunek 2.3.: Muzyka zapisana w pliku MIDI

Warto zaobserwować, że zapis pliku w formacie MIDI umożliwia zdecydowanie większą ekspresję, ponieważ jest to zapis odtworzenia przez artystę pewnego utworu. Ten sposób nie ogranicza muzyki sztywno do konkretnego rytmu zdefiniowanego przez autora.

### 2.1.3. Notacja ABC

Notacja ABC jest systemem zapisywania nut w postaci czystego tekstu znakami ASCII. Notacja pojawiła się w latach 70 XX wieku w celu zapisu i nauki tradycyjnych irlandzkich melodii, a następnie w kolejnej dekadzie została rozwinięta przez Chrisa Walshawę, który zapisywał w niej tradycyjne melodie zanim nauczył się standardowego zachodniego zapisu nutowego. Zapis ten posłużył do stworzonego przez niego programu *abc2mtex*, który na podstawie notacji ABC generował komendy pozwalające zapis partytur w postaci *MusicTex*. Obecnie używanym standardem jest wersja z roku 2011.

```
1 X:1
2 T:Chorał no.7
3 A:J.S. Bach
4 Q:1/4=120
5 V:1
6 L:1/16
7 M:4/4
8 K:C clef=G2
9 D4F4G4A4|d4c6B2A2B2|c4B8A4|d12^c4|
10 V:2
11 L:1/16
12 M:4/4
13 K:C clef=G2
14 A,4D6E2F4|A8^G4A2^G2|A6^G^F^G4E4|A6G2B4A4|
15 V:3
16 F,4A,4^A,4D4|F4E7DC2D2|E2F2B,2E4D2^C4|D6E2F4E4|
17 V:4
18 L:1/16
19 M:4/4
20 K:C clef=F4
21 D,8G,,4D,,4|D,4A,4E,4F,4|C,2D,2E,4E,,4A,,4|F,4^A,4A,2^G,2A,4|
```

Rysunek 2.4.: Zapis wielu głosów w notacji ABC.

Jak widać notacja ABC w dość zwięzły sposób zapisuje partyturę. Poza konkretnymi nutami oraz rytmem w tym formacie możemy zapisać również dodatkowe informacje na temat utworu. Każda linijka zaczynająca się od znaku A-Z a następnie dwukropkiem jest tak zwanym “ polem informacyjnym”. W tych polach zapisywać można takie informacje jak tytuł utworu lub metrum oraz wiele innych informacji między innymi dotyczące z jakiego zbioru

muzycznego pochodzi dany utwór. Wiele z tych informacji powinna być usunięta w procesie preprocessingu danych, tak aby model dostał tylko te informacje, które rzeczywiście pomogą w nauce struktury muzyki. Do takich pól należy przede wszystkim domyślna długość nuty (L:), metrum (M:) oraz tonacja (K:) w jakiej utwór został napisany. Każda z tych informacji jest możliwa do "odgadnięcia" przez model, jednak podając modelowi te informacje wyraźnie, mamy większą kontrolę nad procesem treningu. Dodatkową zaletą podawania tych informacji podczas treningu jest możliwość podania ich jako początkowa sekwencja na podstawie której model dalej będzie próbował kończyć melodię, przez co mamy kontrolę np. nad tym w jakiej tonacji i w jakim metrum zostanie wygenerowany nasz utwór.

Notacja ABC wpiera również melodie polifoniczne przy pomocy tagów V. Ich ilość nie jest ograniczona w związku z czym można w tym zapisie jest możliwe ujęcie nawet muzyki orkiestrowej. Dość skrajnym przykładem jest zapis drugiej części VII symfonii Ludwiga van Beethovena, która składa się aż z 19 instrumentów [2].

### 2.1.3.1. Tokenizacja plików MIDI

Token jest odrębnym elementem, częścią sekwencji tokenów. W języku naturalnym tokenem może być znak, zaimek lub słowo. Zdanie może być następnie tokenizowane na sekwencję tokenów reprezentujących słowa i znaki interpunkcyjne. W przypadku muzyki tokeny mogą reprezentować wartości atrybutów nut (wysokość, wartość, czas trwania) lub zdarzenia czasowe. Token może przyjmować jedną z trzech form:

- nazwa tokenu - słowna reprezentacja *eventu* MIDI np. *Pitch\_50*
- id - unikalna wartość liczbowa przypisana konkretnemu zdarzeniu
- bajt - unikalny bajt, który został przydzielony podczas treningu tokenizera

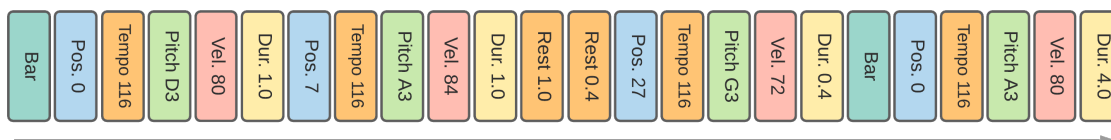
Słownictwo jest zapisywane w postaci *look up table* łączącej nazwę tokenu z odpowiadającym jej id lub bajtem. Trening tokenizera polega na obliczeniu kodowania gramatykowego np. *byte pair encoding* do postaci tabelarycznej w celu wykorzystania ich w dalszym modelowaniu.

Poza tokenami, które tokenizer tworzy na podstawie pliku MIDI dodaje on również dodatkowe takie jak:

- PAD (*padding*) - token używany w przypadku kiedy w *batchu* długość sekwencji jest różna; w takim przypadku tym tokenem wydłuża się sekwencje aby wszystkie miały długość najdłuższej
- BOS (*beginning of section*) - token oznaczający początek sekwencji
- EOS (*end of section*) - token oznaczający koniec sekwencji



Rysunek 2.5.: Prosta sekwencja muzyczna



Rysunek 2.6.: Reprezentacja muzyki w postaci tokenów

Istnieje wiele algorytmów tokenizacji MIDI, jednak aby przedstawić mechanizm działania przyjrzymy się algorytmowi REMI. Revamped MIDI [3] reprezentuje nuty jako sekwencja tokenów wysokości tonu, mocy i długości trwania oraz czas przy pomocy tokenu taktu oraz pozycji. Token taktu oznacza początek nowego taktu, a pozycji jego miejsce w obecnym takcie. Porównanie pomiędzy zapisem nutowym a jego reprezentacją można zaobserwować na grafice 2.5 i 2.6. Wiele algorytmów jest zaimplementowana w bibliotece MidiTok dla języka Python [4].

### 2.1.4. Porównanie zapisu muzycznego

W tabeli 2.1 zostało przedstawione porównanie zapisu całego utworu, którego fragment przedstawiono na grafice 2.1.

	rozmiar pliku na dysku	długość sekwencji
wav	3971116 B	992768
mp3	360951 B	992768
mid	1638 B	740
abc	953 B	562

Tabela 2.1.: Porównanie różnych zapisów muzyki

Jak widać zapis w postaci fali dźwiękowej zajmuje najwięcej miejsca na dysku, jak i również długość sekwencji, która zapisuje cały utwór jest najdłuższa. W celu uzyskania sekwencji z pliku MIDI został użyty tokenizer REMI o którym jest mowa w rozdziale 2.1.3.1. Przy użyciu innego tokenizera niż REMI, liczba powstałych tokenów może lekko różnić się od tej w tabeli. Różnica w rozmiarze na dysku oraz długością sekwencji pomiędzy zapisem MIDI oraz

ABC nie jest widoczna, jednak oba te formaty są zdecydowanie lepszymi rozwiązaniami w przypadku generacji notacji muzycznej (dziwne zdanie do poprawy raczej). Warto zwrócić uwagę, że gdyby w oryginalnym utworze pojawiły się powtórzenia danych sekwencji melodii notacja ABC byłaby jeszcze bardziej oszczędna w obu przypadkach od plików MIDI. Powodem tego jest fakt, że notacja ABC zapisuje bezpośrednio w postaci tekstowej zapis nutowy, czyli znak powtórzenia zostanie zapisany jako `:|` oraz `|:`, a w pliku MIDI cała powtarzana sekwencja musi zostać ponownie odtworzona przez artystę.

Do głównej zalety plików MIDI należy ich wszechstronność. Nie są one ograniczone do kodowania sztywno określonego rytmu tak jak w notacji muzycznej. Dodatkowo jest bardzo łatwo przy użyciu narzędzi komputerowych zamienić dowolny zapis nutowy (MusicXML, ABC) na plik MIDI. Niestety w drugą stronę nie jest już tak łatwo, właśnie w związku z nie konkretnym rytmem, który może zostać zakodowany w MIDI. Istnieją narzędzia takie jak np. MuseScore, które jest w stanie wygenerować partyturę z pliku MIDI, jednak nie jest ono w każdym przypadku jednakowo skuteczne.

## 2.2. Zbiory danych

### 2.2.1. Johann Sebastian Bach Chorales

Dataset [5]

### 2.2.2. The MAESTRO v3.0

Dataset [6]

### 2.2.3. Million Song Dataset

Dataset i takie cytowanko [7]

## 2.3. STOA

Tutaj nie wiem do końca w jakiej kolejności chciałbym o tym pisać, ponieważ z jednej strony przedstawienie STOA przed czymkolwiek jest ok, ale nie chciałbym pisać o czymś czego jeszcze w pracy nie wprowadziłem.

## **2.4. Architekturze transformera**

### **2.4.1. Algorytm uwagi (ang. *attention*)**

### **2.4.2. Warianty mechanizmu uwagi**

#### **2.4.2.1. Self attention**

#### **2.4.2.2. Multi-headed attention**

#### **2.4.2.3. Flash attention**

### **2.4.3. Budowa transformera**

### **2.4.4. Modele tranformerowe**

#### **2.4.4.1. *Classic* transformer**

#### **2.4.4.2. SeqGAN**

#### **2.4.4.3. Mistral**

## **2.5. Architektura *state space***

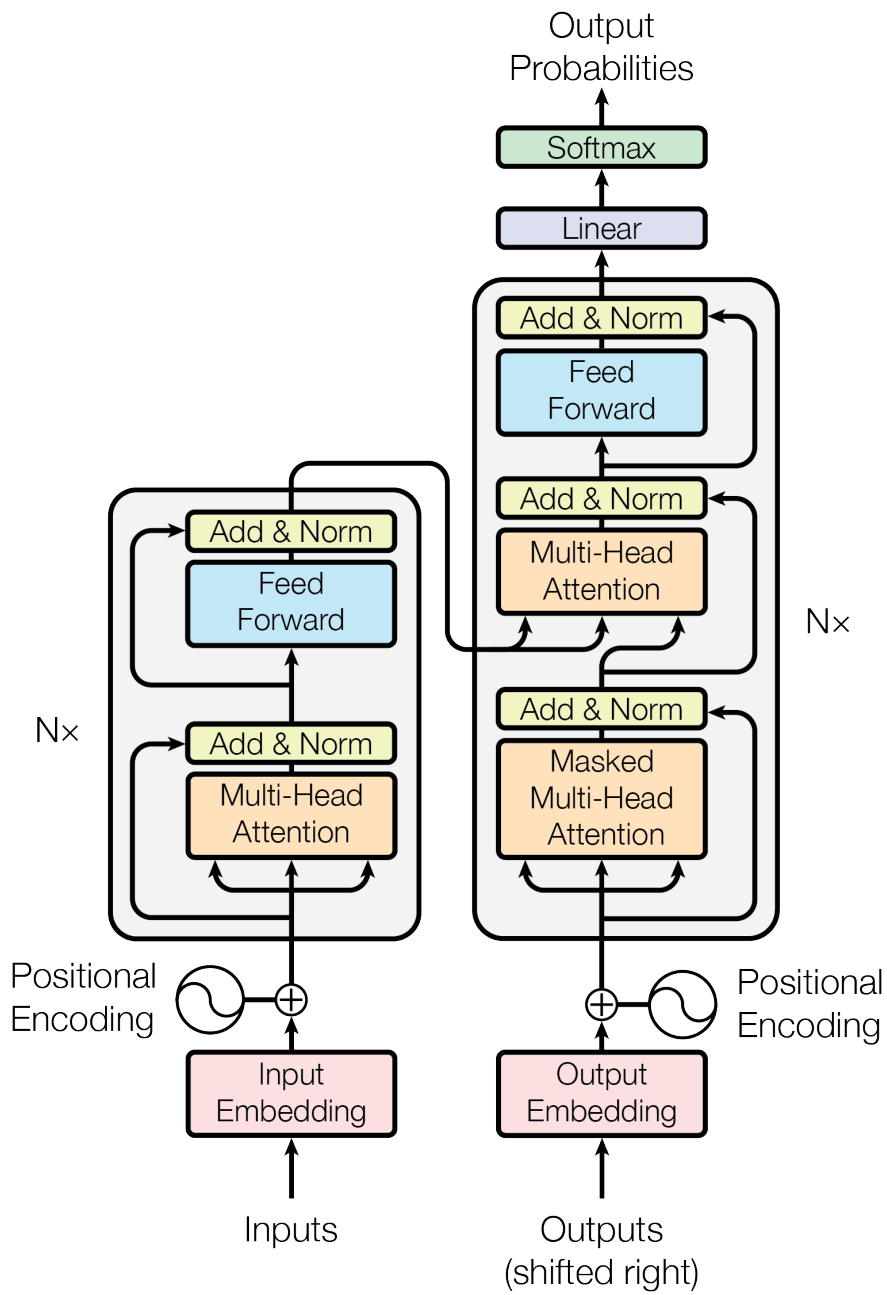
### **2.5.1. Mamba**

### **2.5.2. Tutaj się rozdrobnić trzeba**

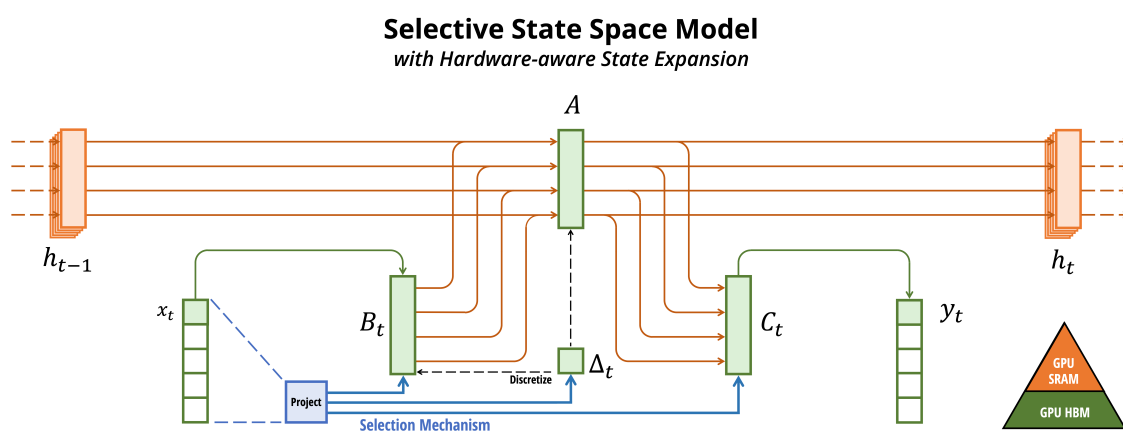
**Uwaga 2.1.** Tytuł oraz strukturę rozdziału należy ustalić z opiekunem pracy.

Aktualny stan wiedzy, na dany temat, na podstawie dostępnej literatury naukowej oraz specjalistycznej.





Rysunek 2.7.: Schemat transformera.



Rysunek 2.8.: Schemat modelu Mamba.

### **3. Propozycja rozwiązania**



## 4. Przeprowadzenie eksperymentów

**Uwaga 4.1.** Tytuł oraz strukturę rozdziału należy ustalić z opiekunem pracy.

### 4.1. Opis *pipeline-u*

Tutaj zamierzam opisać w jaki sposób modele zostały stworzone, jakie biblioteki zostały użyte, jaki sprzęt został użyty podczas treningu

### 4.2. Porównanie architektur użytych modeli

### 4.3. Prezentacja otrzymanych wyników

### 4.4. Porównanie wyników

Celem porównania otrzymanych wyników z dowolnych modeli, jest ocena jakości ich pracy. W kontekście modeli dyskryminujących (ang. *discriminative model*), na przykład klasyfikatorów bądź modeli regresyjnych, ocena ich jakości jest dość prosta, ponieważ istnieje predefiniowana, prawdziwa wartość w zbiorze testowym, którą model próbuje przewidzieć. W takim przypadku ewaluacja jakości modelu to porównanie prawdziwych danych, z tymi przewidzianymi przez model. Porównanie odbywa się przez policzenie metryk np. dla modelu klasyfikującego celności, precyzji, *F1-score* lub dla modelu regresyjnego MSE lub  $R^2$ . W przypadku modeli generacyjnych celem treningu jest jak najlepsza aproksymacja rozkładu prawdopodobieństwa danych  $P(X_{data})$  lub w przypadku danych oznaczonych łączny rozkład  $P(X, Y)$ . W przypadku dużej wymiarowości danych, obliczenie obiektywnych metryk takich jak *log-likelihood* lub dywergencja Kullbacka-Leiblera (*KLD*) często jest nieobliczalne. Dla człowieka weryfikacja wyników modeli generacyjnych takich jak *text-to-speech* lub *text-to-image* jest trywialnym zadaniem, jednak nie jest to oczywiste zadanie algorytmiczne. Często odwołuje się do subiektywnych metryk takich jak *MOS* ang. *mean opinion score*, które oblicza się jako średnią opinię np. w skali od 1-5 braną z zazwyczaj niewielkiej grupy ludzi. Niestety metryka ta często nie jest wiarygodna ze względu na jej subiektywności niewielką

próbę badawczą. Aby wyeliminować te wady serwisy takie jak *HuggingFace* udostępniają narzędzia dla członków społeczności, które pozwalają na ranking modeli, z nadzieją że zgodnie z prawem wielkich liczb, przy wystarczającej liczbie odpowiedzi, uda się otrzymać w miarę obiektywną ocenę. Przykładowym narzędziem tego rodzaju jest *The TTS Arena*[8], która pozwala na ranking modeli *text-to-speech*.

W przypadku muzyki, istnieje możliwość aby zweryfikować poprawność wygenerowanych sekwencji odwołując się do teorii oraz harmonii muzyki. Istnieje kilka narzędzi takich jak *Chordify*, *Hooktheory* lub *Sibelius*, które pozwalają na analizę harmoniczną utworów, dzięki czemu autorzy mogą w prosty sposób analizować i dobierać progresję danej melodii. Niestety większość takich narzędzi jest płatna i nie pozwala na zautomatyzowaną analizę wielu plików. Dodatkowym problemem jest w analizie harmoniczej, szczególnie prowadzonej przez algorytmy, jest rozróżnienie harmonii wertykalnej oraz horyzontalnej. Rozróżnienie to zostało wytłumaczone przez Jacoba Colliera, kilkukrotnego laureata nagród *Grammy*, którego zdaniem akord, który nawet zagrany sam brzmi niepoprawnie, w odpowiednim kontekście i przez odpowiednią progresję w kolejnych fragmentach muzyki, może mieć nadany sens, przez co cała sekwencja nabiera muzycznego piękna[9].

**Uwaga 4.2.** W mojej pracy prawdopodobnie zostanie zastosowane podejście MOS dla dość niewielkiej grupy ludzi, jednak jeśli triale oprogramowania pozwolą, spróbuję przynajmniej sprawdzić czy taka analiza pozwala na jakieś sensowne obliczenie metryki

Brak obiektywnych metryk, którymi można się posłużyć podczas ewaluacji modelu jest dodatkowym problemem w momencie *fine-tuningu* modelu. Ciężko jest wybrać odpowiednie rozwiązanie architektoniczne lub odpowiednie hiperparametry, kiedy jedyną miarą jakości wygenerowanych danych jest subiektywna opinia programisty lub grupy osób wybranych jako wyrocznia.

## 5. Zakończenie

**Uwaga 5.1.** Tytuł oraz strukturę rozdziału należy ustalić z opiekunem pracy.

1. Podsumowanie.
2. Możliwości dalszego rozwoju.
3. Potencjalne obszary zastosowania pracy.





# Dodatek A.

## Typowe elementy składowe pracy dyplomowej z informatyki

### A.1. Tabele

Uwaga A.1.

- Każda tabela powinna być opisana w treści pracy.
- Podpis ma być przed tabelą.

W tabeli [A.1](#) przedstawiono wyniki pomiarów.

Tabela A.1.: Pomiary zużycia energii elektrycznej.

L.p.	Wartość
1	12345,6789
	45,89
2	45,678901

Jeżeli tabela zawiera dużą liczbę wierszy i może nie zmieścić się na stronie — patrz tabela [A.2](#) — skorzystaj z pakietu *longtable* [\[10\]](#).

Tabela A.2.: Tabela, która zawiera dużą liczbę wierszy.

	1	2	3	4	5	6	7	8	
Student 1									

	1	2	3	4	5	6	7	8	
Student 2									
Student 3									
Student 4									
Student 5									
Student 6									
Student 7									
Student 8									
Student 9									

Tabele, w których występuje długi tekst, a co za tym idzie może się on nie zmieścić — musi zostać zawinięty, z pomocą przychodzi środowisko 'tabularx' [11] — patrz tabela A.3.

Tabela A.3.: Tabela zawierająca długi tekst.

Wpis wielokolumnowy!		TRZY	CZTERY
jeden	Szerokość tej kolumny zależy od szerokości tabeli.	trzy	Kolumna czwarta będzie zachowywać się w taki sam sposób jak druga kolumna o tej samej szerokości.

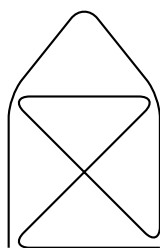
## A.2. Rysunki

### Uwaga A.2.

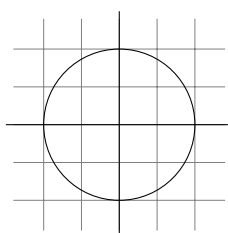
- Rysunki powinny być przerysowane samodzielnie albo używane tylko te, których twórcy zezwolili na ich rozpowszechnianie oraz kopiowanie, czyli np. rysunki objęte licencją Creative Commons.
- Każdy rysunek powinien być opisany w treści pracy.

### A.2.1. Wewnętrzne

Klasa *agh-wi*, automatycznie, dołącza pakiet *TikZ* [12] — dostarcza on komend pozwalających na tworzenie grafik. Przykładowe grafiki pokazano na rysunku A.1 oraz A.2.



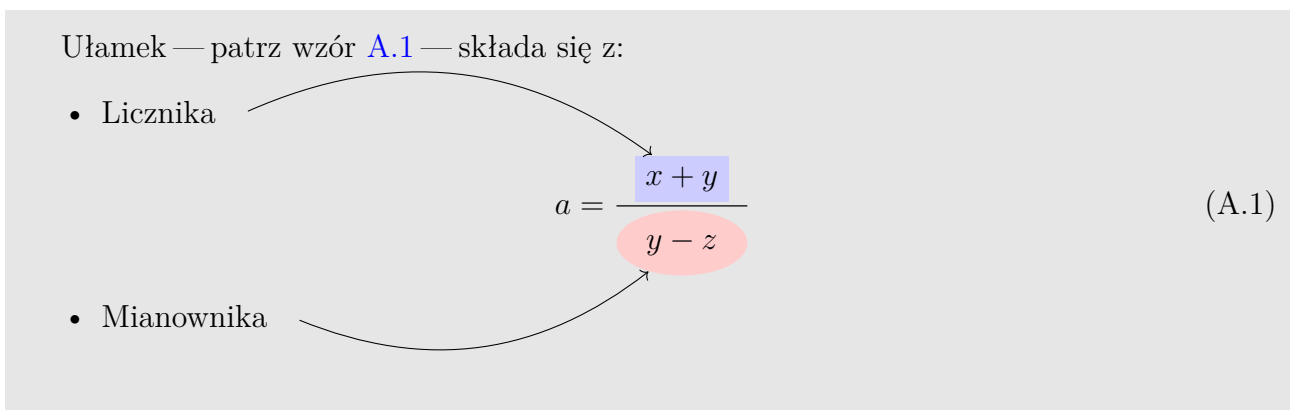
Rysunek A.1.: Prosty rysunek *TikZ*.



Rysunek A.2.: Bardziej złożony rysunek *TikZ*.

Oprócz rysunków eksponowanych możliwe jest tworzenie grafik będących  częścią zdania.

*TikZ* pozwala również na kreślenie po powierzchni strony, np. możemy narysować strzałki pomiędzy elementami strony.



### A.2.2. Zewnętrzne

Oczywiście możliwe jest również dołączanie rysunków zewnętrznych — pakiet *graphicx* [13] pozwala na wstawianie grafik zapisanych w plikach: '.png', '.jpg' oraz '.pdf'. Rysunek A.3 wstawiono przy użyciu tego pakietu.



Rysunek A.3.: Logo Wydziału Informatyki.

## A.3. Kody źródłowe

Najpopularniejszymi pakietami, które umożliwiają składanie kodów źródłowych programów, są:

**listings** [14] — kod źródłowy jest formatowany bezpośrednio przez  $\text{\LaTeX}$ -a — nie jest używany żaden, zewnętrzny, formater kodu.

Kod źródłowy A.1: Przykładowy kod źródłowy sformatowany za pomocą pakietu 'listings'.

```

1  /* Pierwszy program w C++ */
2
3  #include <iostream>
4
5  int main() {
6  std::cout << "Hello World!";
7  return 0;
8  }
```

**minted** [15] — formatuje kod źródłowy przy użyciu biblioteki języka Python o nazwie *Pygments* [16].

Kod źródłowy A.1.: Przykładowy listing sformatowany za pomocą pakietu 'minted'.

```

1  /* Pierwszy program w C++ */
2
3  #include <iostream>
4
5  int main() {
6  std::cout << "Hello World!";
7  return 0;
8  }
```

### Uwaga A.3.

- Podpis ma być przed kodem źródłowym.
- **Proszę używać tylko jednego z tych pakietów**; w przeciwnym razie otrzymasz taki efekt, jak w przykładowej pracy — obydwie listingi mają ten sam numer.

Kod źródłowy w C++ sformatowany przy użyciu pakietu *listings*, pokazano na listingu A.1; sformatowany przy użyciu pakietu *minted*, pokazano na listingu A.1.

## A.4. Algorytmy

Pakiet *algorithm2e* [17] to jeden z kilku, które pozwalają zapisywać algorytmy w formie pseudokodu — patrz algorytm 1.

**Uwaga A.4.** Podpis ma być przed algorytmem.

---

**Algorytm 1:** Disjoint decomposition.

---

```
input : A bitmap  $Im$  of size  $w \times l$ 
output: A partition of the bitmap
1 special treatment of the first line;
2 for  $i \leftarrow 2$  to  $l$  do
3   special treatment of the first element of line  $i$ ;
4   for  $j \leftarrow 2$  to  $w$  do
5      $\text{left} \leftarrow \text{FindCompress}(Im[i, j - 1]);$ 
6      $\text{up} \leftarrow \text{FindCompress}(Im[i - 1,]);$ 
7      $\text{this} \leftarrow \text{FindCompress}(Im[i, j]);$ 
8     if  $\text{left}$  compatible with this then //  $0(\text{left}, \text{this}) == 1$ 
9       if  $\text{left} < \text{this}$  then  $\text{Union}(\text{left}, \text{this});$ 
10      else  $\text{Union}(\text{this}, \text{left});$ 
11    end
12    if  $\text{up}$  compatible with this then //  $0(\text{up}, \text{this}) == 1$ 
13      if  $\text{up} < \text{this}$  then  $\text{Union}(\text{up}, \text{this});$ 
14      // this is put under up to keep tree as flat as possible
15      else  $\text{Union}(\text{this}, \text{up});$ 
16      // this linked to up
17    end
18  end
19 foreach element  $e$  of the line  $i$  do  $\text{FindCompress}(p);$ 
20 end
```

---

## A.5. Wzory

L<sup>A</sup>T<sub>E</sub>X bardzo dobrze sprawdza się w przypadku prac dyplomowych zawierających wzory matematyczne<sup>1</sup>.

---

<sup>1</sup>W przypadku złożonych wzorów warto zastosować pakiet *amsmath* [18].

### A.5.1. Przykłady

Wzór  $E = mc^2$  jest częścią zdania.

$$\left| \sum_{i=1}^n a_i b_i \right| \leq \left( \sum_{i=1}^n a_i^2 \right)^{1/2} \left( \sum_{i=1}^n b_i^2 \right)^{1/2} \quad (\text{A.2})$$

Wartości zmiennej opisano wzorem A.3.

$$x = \begin{cases} y & \text{dla } y > 0 \\ \frac{z}{y} & \text{dla } y \leq 0 \end{cases} \quad (\text{A.3})$$

Wzór A.4 to wzór wielowierszowy.

$$\begin{aligned} 2x^2 + 3(x-1)(x-2) &= 2x^2 + 3(x^2 - 3x + 2) \\ &= 2x^2 + 3x^2 - 9x + 6 \\ &= 5x^2 - 9x + 6 \end{aligned} \quad (\text{A.4})$$

**Uwaga A.5.** Należy używać tylko dwóch rodzajów wzorów:

1. „W linii”.
2. Eksponowane, numerowane.

## A.6. Twierdzenia i podobne struktury

Twierdzenie nr 1 opublikował, w roku 1691, francuski matematyk Michel Rolle.

**Twierdzenie 1 (Rolle’a)** *Jeśli dana funkcja  $f: \mathbb{R} \rightarrow \mathbb{R}$  jest:*

1. ciągła w przedziale  $[a, b]$
2. jest różniczkowalna w przedziale  $(a, b)$
3. na końcach przedziału  $[a, b]$  przyjmuje równe wartości:  $f(a) = f(b)$ ,

*to w przedziale  $(a, b)$  istnieje co najmniej jeden punkt  $c$  taki, że  $f'(c) = 0$ .*

Teraz coś z informatyki ...

**Definicja 1** *Bit to najmniejsza jednostka informacji w komputerze.*

**Definicja 2** *Bajtem nazywamy ciąg ośmiu bitów.*





# Uwagi Autora

- Aktualna wersja klasy jest dostępna pod adresem <https://github.com/polaksta/LaTeX/tree/master/agh-wi><sup>1</sup>.
- Skoro Twoja praca dyplomowa powstała w L<sup>A</sup>T<sub>E</sub>Xu, to zachęcam Cię również do przygotowania prezentacji (na obronę pracy magisterskiej) w tym języku. Najpopularniejszą klasą do tworzenia tego typu dokumentów jest *beamer* [19].
- Pod adresem <https://github.com/polaksta/LaTeX/tree/master/beamerthemeAGH><sup>2</sup> możesz znaleźć, stworzony przeze mnie, nasz uczelniany szablon dla prezentacji L<sup>A</sup>T<sub>E</sub>X Beamer.
- Treść wszystkich rozdziałów tej, przykładowej, pracy dyplomowej znajduje się w jednym pliku — **nie jest to polecane rozwiązanie**. W przypadku pisania własnej pracy warto umieścić zawartość każdego z rozdziałów w osobnych plikach, a następnie dołączać je do dokumentu głównego — patrz opis na stronie <https://www.dickimaw-books.com/latex/thesis/html/include.html>.
- Jeżeli pewne elementy mają być wyróżniane w **jednakowy** **sposób**, to proponuję nie używać bezpośredniego stylowania, tzn.

```
1 \colorbox{red!50}{jednakowy} \colorbox{red!50}{sposób}
```

ale zdefiniować własną komendę stylującą, np. `\alert`,

```
1 \newcommand{\alert}[1]{\colorbox{red!50}{#1}}
```

a następnie użyć jej w dokumencie.

```
1 \alert{jednakowy} \alert{sposób}
```

Dzięki temu, jeżeli będziesz chciał / chciała zmienić sposób stylowania tych elementów, np. niebieskie tło zamiast czerwonego, to wystarczy zmodyfikować, tylko, definicję komendy, zamiast zastępować, w tekście pracy dyplomowej, wybrane (niekoniecznie wszystkie!) wystąpienia tekstu `red`, tekstem `blue`.

<sup>1</sup>W przypadku Overleaf-a jest ona pod adresem <https://www.overleaf.com/read/fnvcvqjyrbyw#5ac622>

<sup>2</sup>W przypadku Overleaf-a jest on pod adresem <https://www.overleaf.com/read/fkjdtbnbrfhj#9c6184>

Stanisław Polak

# Bibliografia

- [1] Ashish Vaswani i in. *Attention Is All You Need*. 2023. arXiv: [1706.03762](https://arxiv.org/abs/1706.03762) [cs.CL].
- [2] Steve Allen. *Beethoven Symphony No. 7, Movement 2 in ABC*. URL: <https://www.ucolick.org/~sla/abcmusic/sym7mov2.html>.
- [3] Yu-Siang Huang i Yi-Hsuan Yang. „Pop Music Transformer: Beat-based Modeling and Generation of Expressive Pop Piano Compositions”. W: *Proceedings of the 28th ACM International Conference on Multimedia*. MM '20. Seattle, WA, USA: Association for Computing Machinery, 2020, s. 1180–1188. ISBN: 9781450379885. DOI: [10.1145/3394171.3413671](https://doi.org/10.1145/3394171.3413671). URL: <https://doi.org/10.1145/3394171.3413671>.
- [4] Nathan Fradet i in. „MidiTok: A Python package for MIDI file tokenization”. W: *Extended Abstracts for the Late-Breaking Demo Session of the 22nd International Society for Music Information Retrieval Conference*. 2021. URL: <https://archives.ismir.net/ismir2021/latebreaking/000005.pdf>.
- [5] Darrell Conklin. *Bach Chorales*. UCI Machine Learning Repository. DOI: <https://doi.org/10.24432/C5G>
- [6] Curtis Hawthorne i in. „Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset”. W: *International Conference on Learning Representations*. 2019. URL: <https://openreview.net/forum?id=r1lYRjC9F7>.
- [7] Thierry Bertin-Mahieux i in. „The Million Song Dataset”. W: *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*. 2011.
- [8] *The TTS Arena*. URL: <https://huggingface.co/blog/arena-tts>.
- [9] Jacob Collier. *That’s not a wrong note, you just lack confidence*. URL: [https://www.youtube.com/watch?v=meha\\_FCcHbo](https://www.youtube.com/watch?v=meha_FCcHbo).
- [10] *The longtable package*. URL: <http://mirrors.ctan.org/macros/latex/required/tools/longtable.pdf>.
- [11] *The tabularx package*. URL: <http://mirrors.ctan.org/macros/latex/required/tools/tabularx.pdf>.
- [12] *The TikZ and PGF Packages*. URL: <http://mirrors.ctan.org/graphics/pgf/base/doc/pgfmanual.pdf>.
- [13] *Packages in the ‘graphics’ bundle*. URL: <http://mirrors.ctan.org/macros/latex/required/graphics/grfguide.pdf>.

- [14] *The Listings Package*. URL: <http://mirrors.ctan.org/macros/latex/contrib/listings/listings.pdf>.
- [15] *The minted package: Highlighted source code in L<sup>A</sup>T<sub>E</sub>X*. URL: <http://mirrors.ctan.org/macros/latex/contrib/minted/minted.pdf>.
- [16] *Strona WWW biblioteki „Pygments”*. URL: <https://pygments.org/>.
- [17] *algorithm2e.sty — package for algorithms*. URL: <http://mirrors.ctan.org/macros/latex/contrib/algorithm2e/doc/algorithm2e.pdf>.
- [18] *User’s Guide for the amsmath Package*. URL: <http://mirrors.ctan.org/macros/latex/required/amsmath/amslatex/doc/amsldoc.pdf>.
- [19] *The beamer class*. URL: <http://mirrors.ctan.org/macros/latex/contrib/beamer/doc/beameruserguide.pdf>.