POLITECHNIKA ŚLĄSKA

WYDZIAŁ AUTOMATYKI, ELEKTRONIKI I INFORMATYKI

Praca dyplomowa inżynierska

Tworzenie muzyki za pomocą sztucznych sieci neuronowych

autor: Grzegorz Kazana

kierujący pracą: dr inż. Grzegorz Baron

Gliwice, listopad 2019

załącznik nr 2 do zarz. nr 97/08/09

Oświadczenie

Wyrażam zgodę / Nie wyrażam zgody* na udostępnienie mojej pracy dyplomowej / rozprawy doktorskiej*.

Gliwice, dnia 27 listopada 2019

.....
(podpis)

.....
(poświadczenie wiarygodności
podpisu przez Dziekanat)

* podkreślić właściwe

Oświadczenie promotora

Oświadczam, że praca „Tworzenie muzyki za pomocą sztucznych sieci neuronowych” spełnia wymagania formalne pracy dyplomowej inżynierskiej.

Gliwice, dnia 27 listopada 2019

.....

(podpis promotora)

Spis treści

Rozdział 1

Wstęp

Współczesne zastosowania metod uczenia maszynowego są bardzo szerokie. Z pośród niezliczonej ilości problemów praktycznie niemożliwych do rozwiązania podejściami klasycznymi, duża część zostaje rozwiązana z wykorzystaniem szerokiej gamy algorytmów oraz modeli uczenia.

Zadania stawiane przed systemami sztucznej inteligencji można podzielić między innymi na problemy:

- klasyfikacji - polegające na przypisaniu przykładu do określonej kategorii, np. rozpoznawanie obiektów na obrazie
- grupowania - polegające na wyodrębnieniu przypadków o podobnych cechach lub współwystępujących, np. systemy sugerujące przedmioty w sklepach internetowych
- regresji - polegające na oszacowaniu wartości ciągłej dla poszczególnych przykładów, np. przewidywanie cen produktów.

Pomimo doniosłości oraz znaczenia zastosowań powyższych zagadnień, nie można w nich dostrzec przejawów kreatywności. Jednym z problemów, w którym można doszukiwać się inwencji jest tworzenie muzyki.

1.1 Cel i zastosowania

Celem pracy jest analiza i porównanie wyników różnych podejść do obróbki plików muzycznych oraz sposobów syntezy próbek z wykorzystaniem metod uczenia maszynowego.

Pomimo że praca ma charakter badawczo-eksperymentatorski, opracowane metody oraz ich wyniki mogą potencjalnie znaleźć zastosowanie w zadaniach ekstrakcji stylu, augmentacji istniejących zbiorów danych, lub w optymistycznym scenariuszu jako narzędzie wspierające proces twórczy muzyków.

1.2 Zakres pracy

Do zadań autora pracy należy

- dobór zbioru danych
- dobór, implementacja i analiza metod przetwarzania plików muzycznych
- dobór i parametryzacja architektury modelu uczenia maszynowego
- wykorzystanie przygotowanych danych w procesie uczenia i syntezy nowych próbek
- ocena uzyskanych wyników

Rozdział 2

Analiza tematu

Celem zadania doboru i analizy metod obróbki plików muzycznych jest przekształcenie danych do postaci użytecznej przez algorytmy uczenia maszynowego. Do istotnych aspektów tego kroku należy między innymi stopień kompresji danych, sposób wyrażania relacji między przykładami, możliwość bezstratnej transformacji odwrotnej.

Zadanie generowania muzyki polega na ekstrakcji pewnych cech charakterystycznych przykładowych utworów, np. stylu konkretnego artysty, i wykorzystaniu ich przy syntezie tworzonych próbek.

2.1 Wprowadzenie do dziedziny

Jedną z najbardziej rozpowszechnionych metod uczenia maszynowego jest zastosowanie sieci neuronowych. Zaproponowane w latach czterdziestych i rozwijane w drugiej połowie XX wieku, sztuczne sieci neuronowe czerpią inspirację z sposobu funkcjonowania ludzkiego mózgu zbudowanego z komórek nerwowych - neuronów. Połączenia między komórkami są modelowane poprzez wagi, reprezentujące siłę połączenia, a zjawisko aktywacji komórek w sieci poprzez operację ważonej sumy informacji z połączonych neuronów oraz ich wag. W latach siedemdziesiątych opracowano algorytm uczenia sztucznych sieci z powodzeniem wykorzystywany do dziś - propagację wsteczną. Metoda ta polega na minimalizacji błędu predykcji poprzez regulację wag w oparciu o pochodną funkcji błędu. Zastosowania sieci neuronowych

są szerokie, co potwierdza twierdzenie stanowiące o ich możliwości aproksymacji dowolnej funkcji ciągłej w zamkniętym przedziale.

Mimo tego, że ta metoda uczenia maszynowego jest znana od wielu dekad, dopiero w ostatnich latach przeżywa swoisty renesans spowodowany wzrostem dostępnej mocy obliczeniowej oraz możliwością zastosowania w procesie uczenia kart graficznych znacząco przyśpieszających równoległe operacje matematyczne.

Ponieważ dane reprezentujące muzykę mają postać sekwencji rozłożonej w czasie, konieczne jest wykorzystanie sieci mających możliwość agregacji stanu tj. posiadające pamięć. Sieciami spełniającymi powyższy warunek są modele należące do grupy rekurencyjnych sieci neuronowych. Najprostszym przykładem rekurencyjnej sieci neuronowej jest sieć na której wejście przekazywany jest również stan wyjść z analizy poprzedniego elementu sekwencji. Niestety, taka architektura jest narażona na wiele problemów, takich jak trudność tworzenia powiązań pomiędzy odległymi elementami sekwencji oraz znikający gradient.

Architekturą rozwiązującą powyższe problemy jest zasugerowana przez XXXXXX architektura Long short-term memory network (LSTM). Kosztem jej większych możliwości jest zwielokrotnienie ilości parametrów, co może przekładać się na dłuższy czas uczenia.

2.2 Założenia

Na potrzeby pracy została przyjęta następująca definicja muzyki:

Muzyką nazywamy ciągi dźwięków tworzące kompozycyjną całość.

Utwory muzyczne można analizować pod wieloma względami, takimi jak:

- rytmiczność - organizacja dźwięków w czasie
- melodyczność - sposób zestawiania następujących dźwięków
- harmoniczność - spójność i ład występujący między dźwiękami
- dynamika - zróżnicowanie siły dźwięków

W kontekście pracy skupiono się na dwóch aspektach wynikających z powyższej definicji: rytmiczności i tonalności.

2.3 Przegląd literatury

W podobnej tematyce powstało wiele prac, lecz każda wyróżnia się innym podejściem do problemu.

Elementem wspólnym dużej części prac jest wykorzystany format danych wejściowych. W większości studiowanych artykułów, twórcy decydują się na wykorzystanie danych muzycznych w formacie midi [4,5,6,7], aczkolwiek równie popularnym wyjściem jest skorzystanie z notacji ABC [1,2]. Najrzadsze jest podejście opierające się o wykorzystanie surowych plików audio, np. w formacie wav [8].

Jedną z różnic występujących między podejściami jest sposób obróbki i tworzenia wewnętrznej reprezentacji danych, gdyż praktycznie wszyscy autorzy proponują własne rozwiązanie problemu. W przypadku danych tekstowych w formacie ABC, częstym elementem jest ograniczenie lub usunięcie meta informacji zawartych w plikach [1,2] oraz pominięcie białych znaków. Konkretne znaki zapisu są albo przedstawione jako wektory kodu 1 z N [2], lub przekształcane są kolejne wartości liczb naturalnych [1]. W pracach korzystających z plików midi, częstym podejściem do problemu reprezentacji wysokości dźwięku jest zastosowanie kodu 1 z N [4], lecz większym zróżnicowaniem cechuje się podejście do reprezentacji danych w wymiarze czasu. Duża część podejść opiera się na restrykcji możliwych wartości rytmicznych do dyskretnego zbioru [5,6], podczas gdy stosowane są również podejścia korzystające z wewnętrznej częstotliwości próbkowania plików midi [7].

Kluczowym aspektem wszystkich prac jest dobór modelu uczenia maszynowego. Najbardziej popularnym wyborem są rekurencyjne sieci neuronowe [1,2,3,7], ale opisywane są również inne podejścia, na przykład:

- modele kontekstowe, takie jak Continous Bag-of-Words [1],
- modele generatywno adwersyjne (Generative Adversial Networks) [1,7],
- modele konwolucyjne [4,8]

Aspektem wartym zwrócenia uwagi, są również poczynione założenia i nałożone ograniczenia. Niektóre metody przytoczone przez autorów prac wliczają:

- pominięcie elementów polifonicznych utworów [2,5],

- ograniczenie zbioru danych do utworów będących w ustalonej tonacji i/lub ustalonym metrum [2,6]
- ograniczenie na występujące wartości rytmiczne [5]

2.4 Odniesienie do istniejących prac

Niniejsza praca w kolejnych rozdziałach prezentuje analizę różnych podejść do kolejnych etapów procesu implementacji systemu, którego celem jest generowanie utworów muzycznych. W rozdziale 3. poruszono problem wyboru formatu danych, oraz przedstawiono opis zbioru na którym były przeprowadzane eksperymenty. Przedstawiono potencjalne trudności wiążące się z wyborem poszczególnych źródeł danych. W kolejnym rozdziale omówiono różne podejścia do zagadnienia reprezentacji wybranego formatu danych w postaci numerycznej. Zawiera również opis wad i zalet każdego z podejść. Treścią piątego rozdziału jest przestawienie obranego typu modelu uczenia maszynowego oraz wybranej architektury, razem z opisem procesu uczenia. Rozdział poświęcony generacji próbek zawiera opis przyjętego podejścia oraz sposobów wymuszania procesu generacji. Dalsze rozdziały poświęcone są wnioskom i analizie wyników.

2.5 Opis narzędzi

Podczas implementacji korzystano z narzędzi:

- Google Colab - usługa będąca środowiskiem obliczeniowym przystosowanym do uruchamiania skryptów w języku Python, z dostępem do akceleratorów obliczeniowych w postaci kart graficznych i jednostek tensorowych,
- Tensorflow - biblioteka języka Python służąca do wydajnych obliczeń, z dużym wsparciem dla tworzenia i uczenia głębokich sieci neuronowych. Podczas pracy skorzystano z wysokopoziomowego interfejsu tf.keras, zawierającego predefiniowane architektury i warstwy sieci,
- Jupyter notebook - środowisko uruchomieniowe języka Python, umożliwiające przejrzystą ilustrację wykonywanych operacji,

- mido - biblioteka ułatwiająca otwieranie i przetwarzania plików w formacie midi.

Rozdział 3

Dobór danych

Jednym z kluczowych aspektów każdego procesu analizy danych i uczenia maszynowego jest odpowiedni dobór danych. Decyzje podjęte na tym etapie pociągają za sobą konsekwencje w kolejnych etapach procesu. W zależności od postaci danych wejściowych, określany jest konieczny nakład pracy w procesie ich obróbki. Ponadto, odpowiedni dobór danych ma znaczący wpływ na otrzymywane wyniki.

W przypadku danych muzycznych należy zwrócić między innymi na cechy takie jak:

- dostępność danych treningowych,
- sposób reprezentacji upływu czasu,
- sposób wyrażenia wysokości dźwięku,
- jasne przedstawienie zależności między dźwiękami,
- stopień kompresji,
- złożoność samego formatu oraz możliwość istnienia niepoprawnych reprezentacji.

3.1 Porównanie formatów danych

3.1.1 Pliki audio

Najmniej restrykcyjnymi formatami plików audio są formaty będące zapisem amplitudy fali akustycznej. Przykładami plików tego rodzaju są pliki WAV oraz AIFF. Kluczowym parametrem plików audio jest częstotliwość próbkowania. Od niej zależy zakres możliwych do wyrażenia częstotliwości. Typową wartością dla plików WAV jest 44100Hz, co według twierdzenia Nyquista-Shannona przekłada się na zakres częstotliwości ograniczony wartością 22050Hz, będącą bliską granicy słyszalności ucha ludzkiego.

Ponieważ wysokości dźwięków wyrażone są jedynie poprzez sekwencje zmian amplitudy sygnału, ich ekstrakcja wymagałaby wykorzystania transformaty Fouriera.

Największą zaletą i wadą tego formatu jednocześnie jest jego swoboda. Postać fali akustycznej pozwala na przedstawienie każdego dźwięku będącego w paśmie przenoszenia, za równo złożonych wieloinstrumentowych kompozycji muzycznych, prostych melodii, jak i dźwięków nie podlegających pod definicję muzyki. Prowadzi to do niskiego stopnia kompresji informacji, wymuszającego złożone metody obróbki i skomplikowane modele uczenia.

3.1.2 Notacja ABC

Notacja ABC jest jedną z cyfrowych postaci klasycznego zapisu nutowego. Dane w plikach ABC przedstawione są za pomocą znaków ASCII.

Ponieważ jest to odpowiednik zapisu nutowego, oznacza to że za równo wysokości dźwięków, jak i wartości rytmiczne należą do dyskretnego zbioru. Fakt ten nakłada spore ograniczenie na treść plików, lecz w kontekście muzyki jest to ograniczenia bardzo pomocne.

Poza informacją o samym dźwięku i jego wartości rytmicznej, format zawiera również metadane utworu takie jak autor, sygnatura i tempo.

Dużą zaletą tego formatu jest jego wysoki stopień kompresji, za pomocą relatywnie krótkich ciągów znaków jesteśmy w stanie opisać znaczną część utworu muzycznego. Głównym mankamentem notacji ABC w kontekście niniejszej pracy

jest fakt, że nie wszystkie ciągi znaków są poprawnymi zapisami w notacji ABC. Oznacza to, że pojedyncze błędy w generowanych sekwencjach mogą powodować niepoprawność całego utworu. W przypadku niemożliwości wyuczenia modelu tworzenia całkowicie poprawnych w ciągów wymagałoby walidacji i obsługi poprawiania błędów w zapisie.

3.1.3 Format midi

Pliki w formacie midi są zapisem wiadomości przesyłanych protokołem komunikacyjnym o ten samej nazwie. Każdy plik midi składa się z kanałów oraz ścieżek. Poszczególne kanały reprezentują poszczególne instrumenty biorące udział w nagraniu. Na poszczególnych ścieżkach znajdują się wiadomości reprezentujące zdarzenia w utworze. Do najczęstszych z nich należą: note_on, note_off, set_tempo.

W przypadku wiadomości związanych z rozpoczęciem lub zakończeniem dźwięku dołączony jest również numer dźwięku z zakresu 0-127. Wartości tego parametru odpowiadają kolejnym dźwiękom skali dwunastotonowej poczynając od C-1 i kończąc na G9. Dostępne dźwięki są nadzbiorem dźwięków dostępnych na klawiaturze klasycznego fortepianu.

Dodatkowym parametrem każdej wiadomości jest wartość time, będąca ilością impulsów (ang. 'ticks') które upłynęły od poprzedniej wiadomości. Oznacza to, że długość trwania dźwięku można wyznaczyć poprzez zsumowanie parametrów time wiadomości występujących pomiędzy odpowiadającymi sobie zdarzeniami note_on i note_off.

Możliwa jest konwersja wartości za równo na czas w milisekundach jak i wartości rytmiczne. Stałe potrzebne do tych przekształceń są naczęściej zawarte w meta wiadomościach znajdujących się na początku utworu.

Zaletą formatu midi jest dyskretna reprezentacja wysokości dźwięku oraz pozwalająca na elastyczną interpretację postać upływu czasu.

3.1.4 Podsumowanie

Wszystkie omówione formaty danych mają wady i zalety. Cechą opisującą wszystkie z nich, jest bogata dostępność danych, przez co wybór dokonano na podstawie pozostałych cech. W dalszej części pracy opisywane podejścia będą tyczyć jedynie

plików w formacie midi, na którego wybór zdecydowano się drogą eliminacji. Pliki audio odrzucono przez wymaganą złożoność wstępnej obróbki, a notację ABC z powodu obaw związanych z trudnościami zagwarantowania syntaktycznej poprawności generowanych ciągów znaków.

3.2 Opis wybranego zbioru danych

Do dalszych eksperymentów, opierających się na różnych sposobach tworzenia numerycznej reprezentacji danych wybrałem stosunkowo mały zbiór zawierający utwory z serii gier Pokemon wyprodukowanych na konsolę Nintendo. Zbiór składa się z dziesięciu utworów o średniej długości wynoszącej 90 sekund. Kompozycje zawierają za równo elementy monofoniczne, jak i polifoniczne. Główną motywacją wyboru tego zbioru była jego mała obszerność, co pozwalało na szybsze testowanie metod obróbki danych i procesu uczenia.

Rozdział 4

Sposoby reprezentacji danych

Znaczna część modeli uczenia maszynowego, w tym sieci neuronowe, wymagają danych w postaci numerycznej. Odpowiednio dobierając proces przekształcania danych jesteśmy w stanie nie tylko umożliwić skorzystanie z tych modeli, ale również możemy uwypuklić informacje prawdziwie w nim istotne, co skróci i ułatwi proces uczenia modelu.

Poniższy rozdział zawiera analizę różnych podejść do przedstawionego problemu.

4.1 Reprezentacja dźwięków

Pierwszym z wymiarów naszych danych są wysokości dźwięków. W przypadku plików midi oryginalnie są to liczby z przedziału 0-127. Są to już dane numeryczne, które moglibyśmy teoretycznie wykorzystać jest bezpośrednio do uczenia modelu. W takim przypadku nasze zadanie sprowadziłoby się do problemu regresji, ponieważ próbowalibyśmy oszacować wartość numeryczną, którą następnie trzeba by ograniczyć do liczb z dozwolonego zakresu.

Jedną z wad tego rozwiązania, jest fałszywe przedstawienie relacji między dźwiękami. Dźwięki znajdujące się w bliższym sąsiedztwie byłyby traktowane jako bardziej podobne. Takie założenie w muzyce nie zawsze jest prawdą. Przykładowo, dźwięki 60 i 66 (C4 i F#4), znajdują się relatywnie blisko, mimo tego że występuje między nimi interwał trytonu, będący jednym z najsilniejszych dysonansów

w skali dwunastotonowej. Z drugiej strony, dźwięki 60 i 84 (C4 i C6) dzieli spora odległość, mimo tego że jest to ten sam dźwięk zagrany dwie oktawy wyżej.

Niniejsze problemy można próbować zaadresować poprzez stosowanie poniższych podejść.

4.1.1 Kody 1 z N i M z N

Klasycznym sposobem na rozwiązanie powyższego problemu jest zastosowanie kodu 1 z N. W przypadku informacji o dźwiękach midi, oznaczałoby to wypełnienie wektora 128 zerami, i postawieniem jedynki na pozycji reprezentującej dany dźwięk. Zaletą tego rozwiązania jest możliwość reprezentacji melodii polifonicznych, co przekształciłoby powyższy kod na kod M z N, gdzie M to ilość dźwięków granych jednocześnie w danym momencie sekwencji. Mimo tego, że w taki sposób pozbywany się fałszywych zależności między sąsiednimi dźwiękami, to nie reprezentujemy zależności prawdziwych, jak w przypadku dźwięków odległych o oktawy. Kolejnym problem tego podejścia jest również znaczący wzrost wymiarowości naszych danych i spadek ich kompresji, co przełoży się na wolniejszy proces uczenia.

4.1.2 Wektory zanurzone

Powyższy problem dużej wymiarowości i niemożliwości wyrażenia relacji między obiektami nie jest charakterystyczny tylko i wyłącznie dla analizy dźwięku. Problem ten występuje również w dziale przetwarzania języka naturalnego. Również w tym kontekście każdy wyraz stanowi jeden element z N, gdzie N jest rozmiarem słownika - zbioru wszystkich wyrazów.

Rozwiązaniem cieszącym się dużym powodzeniem jest algorytm Word2Vec zaproponowany w roku 2013 przez ... . Jest to algorytm służący do wyznaczenia ukrytej reprezentacji kodów 1 z N w postaci wektorów wartości ciągłych o dużo mniejszym wymiarze. Metoda opiera się na założeniu twierdzącym że wektory o podobnym znaczeniu występują w sąsiedztwie podobnych lub nawet tych samych wyrazów. Ponieważ to założenie jest również prawdziwe w kontekście muzyki przyjęto użycie algorytmy Word2Vec za podstawne.

Jedyną modyfikacją potrzebną do wykorzystania tej metody do przykładów polifonicznych, było traktowanie całych wielodźwięków jako pojedynczych obiektów

w słowniku.

4.2 Reprezentacja czasu

Drugim wymiarem danych muzycznych jest czas. Ponownie, wartość oznaczająca upływ czasu mogłaby użyta bezpośrednio, ponownie sprowadzając problem do regresji. Dodatkowo, rozsądnym krokiem byłoby znormalizowanie wartości w impulsach midi do arbitralnie wybranej wartości reprezentującą konkretną wartość rytmiczną. Przykładowym mapowaniem mogłoby być przyjęcie wartości 1 dla ćwierćnut.

Głównym mankamentem tego rozwiązania jest spowodowane naturą regresji ryzyko niemożliwości wyuczenia dokładnych wartości rytmicznych, a jedynie oscylowanie między wartościami. W takim przypadku generowana muzyka mogłaby nie sprawiać wrażenia rytmicznej, gdyż dźwięki nie tworzyłyby spójnych taktów.

4.2.1 Próbkowanie

Alternatywną metodą analizy upływu czasu jest próbkowanie. Z przyjętą częstotliwością próbkowania, określałoby się listę aktualnie wybrzmiewających dźwięków i tworzyło macierz o wymiarach t x n, gdzie n to wymiarowość reprezentacji wysokości dźwięku, a t to ilość pobranych próbek w danym fragmencie utworu.

Ponieważ dźwięki zawsze trwają wielokrotność pewnej wartości, potencjalnie łatwiejsze byłoby odtworzenie rytmicznego charakteru muzyki. Wadą tego podejścia jest konieczność doboru częstotliwości próbkowania. Zbyt niskie wartości uniemożliwiłyby precyzyjne przestawienie utworu. Natomiast, wraz z wyborem większej częstotliwości próbkowania, te same dźwięki byłyby reprezentowane przez większą ilość wektorów, co powodowały oddalenie następujących dźwięków w sekwencji i utrudnienie uczenia.

4.2.2 Grupowanie długości dźwięków

Podejściem łączącym zalety obydwu rozwiązań, jest przeprowadzenie grupowania znormalizowanych czasów w odniesieniu do ustalonej wartości rytmicznej. W ten

sposób możliwe byłoby odzwierciedlenie wartości rytmicznych występujących w zbiorze danych i przedstawienie ich w dyskretny sposób bez arbitralnego określania zbioru dozwolonych wartości. Ponownie, po podziale na grupy problem zostanie przekształcony do problemu klasyfikacji 1 z N.

4.3 Podsumowanie

Z powodu zalet i wad przedstawionych w powyższej analizie, zdecydowano się na reprezentacje dźwięków w postaci wektorów zanurzonych, a wybraną reprezentacją czasu zostały dyskretne wartości uzyskane poprzez grupowanie.

Rozdział 5

Uczenie

5.1 Architektura modelu

Otrzymawszy dane w postaci sekwencji par wektorów opisujących dźwięki i ich czas trwania, przystąpiono do projektu architektury modelu. Z powodu dwuelementowej postaci danych zdecydowano się na zaprojektowanie modelu dwuwejściowego i dwuwyjściowego.

Ponieważ charakter wektorów w sekwencji jest różny - wartości ciągłe opisujące dźwięki i kategoryczny kod 1 z N opisujący długość dźwięku - przed złączeniem wektorów postanowiono przekształcić wektor 1 z N przez warstwę gęstą o wymiarze mniejszym niż N. Celem tej operacji było wprowadzenie konieczności przez sieć kompresji informacji, co najprawdopodobniej przełoży się na wyznaczenie pewnej reprezentacji ciągłej kodu.

Następnie wektory zostają złączone, i ich sekwencje trafiają do warstw rekurencyjnej sieci LSTM, w której model ma szanse ekstrahować wiedzę o zależnościach między elementami sekwencji.

Otrzymywane wektory są połączone do osobnych, mniejszych sieci LSTM i następnie poprzez warstwy gęsto połączone stają się osobnymi wyjściami modelu.

Wyjście odpowiedzialne za dźwięki uczone jest funkcją błędu odpowiednią do problemu regresji - błędem średniokwadratowym, a wyjście klasyfikujące długość dźwięku funkcją odpowiednią dla zadania klasyfikacji 1 z N - categorical crossentrophy.

5.2 Dobór parametrów

Podczas procesu uczenia najistotniejszymi parametrami były:

- rozmiar okna, czyli długość uczonych sekwencji,
- ilość przykładów w wiązce, czyli ile sekwencji było uczonych jednocześnie poprzedzając pojedyncze przeprowadzenie algorytmu propagacji wstecznej,
- rozmiar głównej sieci LSTM.

5.3 Proces uczenia

Podczas uczenia modelu zdecydowano się na dynamiczny rozmiar okna, będący liczbą losową z zakresu <15, 50>. Motywacją tego wyboru były testowe uruchomienia modelu potwierdzające zdroworozsądkową intuicję mówiącą, że większe wartości umożliwią uczenie dłuższych powtarzających się motywów, a krótsze lokalne następstwo nut.

Interesującą obserwacją był wpływ ilości przykładów w wiązce. Większe wartości, rzędu 16 lub 32 miały negatywny wpływ na jakość generowanych sekwencji. Mniejsze wartości prowadziły do przetrenowania modelu.

Rozmiar głównej sieci miał spodziewany efekt, nieodpowiednio duży prowadził do przetrenowania. Dla obecnego zbioru danych zdecydowałem się zachować wartość 64 jednostek.

Rozdział 6

Generowanie próbek

6.1 Opis podejścia

6.2 Opisy ziaren

Rozdział 7

Analiza wyników

7.1 Miary i ich definicje

7.2 Wyniki

7.3 Wrażenia subiektywne

Rozdział 8

Wnioski

Bibliografia

Dodatki