

UNIWERSYTET GDAŃSKI
Wydział Matematyki, Fizyki i Informatyki

Michał Jakóbowski

nr albumu: 240 336

**Generowanie treści przy
pomocy rekurencyjnych sieci
neuronowych**

Praca magisterska na kierunku:

INFORMATYKA

Promotor:

dr Włodzimierz Bzyl

Gdańsk 2017

Streszczenie

W pracy zostało opisane działanie oraz wybrane możliwości wykorzystania modelu rekurencyjnych sieci neuronowych autorstwa Andreja Karpathy'ego, udoskonalonego przez Justina Johnsona.[1] Model ten nosi nazwę Torch-rnn i jego kod źródłowy jest dostępny na platformie GitHub. Jest on realizacją wielopoziomowej rekurencyjnej sieci neuronowej Long Short-Term Memory(LSTM) polegającą na generowaniu treści (sekwencji znaków) na podstawie wcześniej zgromadzonej bazy danych treningowych.

W pierwszym rozdziale omówiono zagadnienie sieci neuronowych, zaczynając od fundamentów jakimi była koncepcja neuronu McCullocha-Pittsa poprzez współczesne sieci neuronowe, a kończąc na rekurencyjnych sieciach neuronowych typu LSTM wykorzystywanych w omawianym modelu. W dalszej części skupiono się na przedstawieniu środowiska programistycznego Torch-RNN, szczegółowo opisano sposób instalacji wszystkich wymaganych bibliotek oraz zależności w systemie Ubuntu. Meritum pracy zostało zawarte w rozdziale opisującym eksperymenty związane z procesem treningowym omawianej sieci neuronowej. Przedstawiono tam kilka wybranych możliwości oraz efektów wykorzystania tychże sieci. Eksperymenty dotyczyły generowania tekstów przypominających dzieła literackie Willama Szekspira (pokazano dramat szekspirowski) oraz Henryka Sienkiewicza (powieść w języku polskim), połączenie świętych ksiąg religijnych(Biblii i Koranu) obsługę języka znaczników platformy Wikimedia (popularnonaukowe artykuły z dziedziny Fizyki) oraz generowanie kodu źródłowego w obiektowym języku programistycznym Java. Na koniec porównano czas działania procesu treningowego w zależności od wybranej architektury sprzętowej oraz stopnia skomplikowania wybranego modelu sieci neuronowej. W podsumowaniu autor wyciąga wnioski z przeprowadzonych eksperymentów oraz przedstawia przewidywania na przyszłość związane z możliwościami jakie dają rekurencyjne sieci neuronowe typu LSTM.[2]

Słowa kluczowe

Rekurencyjne sieci neuronowe, LSTM, torch, torch-rnn, CUDA, python, bash

Spis treści

Wprowadzenie	7
1. Rekurencyjne sieci neuronowe (LSTM)	9
1.1. Koncept neuronu	9
1.2. Wielowarstwowe sieci neuronowe	11
1.3. Rekurencyjne sieci neuronowe	13
1.4. Sieci Long Short-Term Memory (LSTM)	14
2. Środowisko torch-rnn	16
2.1. Przygotowanie bibliotek pomocniczych	17
2.2. Środowisko preprocessingu danych	17
2.3. Framework Torch	19
3. Eksperymenty	21
3.1. Dramat szekspirowski - archaizmy języka angielskiego	21
3.2. Powieści Henryka Sienkiewicza	26
3.3. Konkatenacja świętych ksiąg	29
3.4. Artykuł popularnonaukowy serwisu Wikipedia	30
3.5. Kod źródłowy języka Java	34
4. Benchmark	38
4.1. Nvidia CUDA	38
4.2. Testy	40
Zakończenie	42
Bibliografia	44
Spis tabel	46
Spis rysunków	47

Oświadczenie	48
-------------------------------	----

Wprowadzenie

Rozwinięty mózg jest cechą która wyróżnia nas, ludzi spośród innych przedstawicieli świata zwierząt. Mimo, że zwierzęta często bywają od nas większe i silniejsze, to jednak pewne własności naszego mózgu pozwoliły nam osiągnąć nad nimi przewagę. Mowa tu o szeroko rozumianej inteligencji, którą udało nam się rozwinąć w drodze ewolucji. Ewolucja jednak jest procesem rozciągniętym na bardzo długi okres czasu, a ludzkość chcąc się rozwijać w znacznie szybszym tempie wpadła na pomysł wytworzenia sztucznej inteligencji, która to w zamierzeniu ma osiągnąć nasz poziom. Z czasem być może go prześcigając.

Prowadzić to może do ciężkich do przewidzenia konsekwencji, jednak nie tym chciałbym zająć się w tej pracy. Opiszę natomiast techniczne podstawy tego zagadnienia, skupiając się na jednej z cech jaką moim zdaniem powinna posiadać sztuczna inteligencja. Jest to umiejętność tworzenia tekstu (pisania). To między innymi dzięki tej umiejętności ludzkość mogła osiągnąć obecny poziom rozwoju, przekazując teksty z pokolenia na pokolenie mogliśmy kumulować nasze doświadczenia oraz umiejętności.

Moją motywacją do napisanie tej pracy stał się cytat pozostawiony przez jednego z pionierów zagadnienia sztucznej inteligencji jakim był Alan Turing. W roku 1950 zaproponował on pomysł na powołanie do życia sztucznej inteligencji. Chodzi tu o propozycję tak zwanego "umysłu dziecka", którą opisał następująco:

Może zamiast dążyć do wytworzenia programu udającego umysł dorosłego, spróbować raczej wyprodukować program udający umysł dziecka? Poddawszy go stosownej edukacji, uzyskalibyśmy umysł dorosłego.[3]

Ten prosty w swych założeniach pomysł zdaje się przyświecać zagadnieniu sieci neuronowych. Są to składające się z wielu sztucznych neuronów sieci, które w swych założeniach mają imitować działanie prawdziwego mózgu. Owe symulatory modeli matematycznych analizując pewien zbiór danych posiadają zdolność do przyswajania pewnych zależności. Dzięki nim można

próbować rozwiązywać problemy, których nie udałooby się rozwiązać w sposób algorytmiczny (lub wymagałoby to dużych zasobów).

Takim problemem jest właśnie omawiane w tej pracy generowanie tekstu. Do swoich eksperymentów, których przebieg szczegółowo opiszę w tej pracy, wykorzystałem implementację rekurencyjnych sieci neuronowych zaproponowaną przez Andreja Karpathy'ego z Uniwersytetu Stanforda.[\[4\]](#) Większość prac była prowadzona na środowisku sprzętowym udostępnionym przez Instytut Informatyki Uniwersytetu Gdańskiego.

ROZDZIAŁ 1

Rekurencyjne sieci neuronowe (LSTM)

1.1. Koncept neuronu

Sieciami neuronowymi określa się symulatory (programowe lub sprzętowe) modeli matematycznych realizujące pseudorównoległe przetwarzanie informacji, składające się z wielu wzajemnie połączonych neuronów i naśladujące działanie biologicznych struktur mózgowych.^[5]

Zagadnienie to budzi coraz większe zainteresowanie na całym świecie. Z pewnością można je zakwalifikować jako dziedzinę nauk technicznych, jednak warto tutaj wspomnieć o źródłach z których wywodzi się ta nowoczesna technika. Podstaw konceptu sieci neuronowych należy się doszukiwać w odkryciach biologów którzy to śledzą tajemnice ludzkiego mózgu. Wielu z nich doczekało się najważniejszej nagrody w świecie nauki - czyli nagrody Nobla. Poniżej zaprezentowano tabelę z najważniejszymi odkryciami z dziedziny neurologii wraz z datą oraz nazwiskiem odkrywcy:

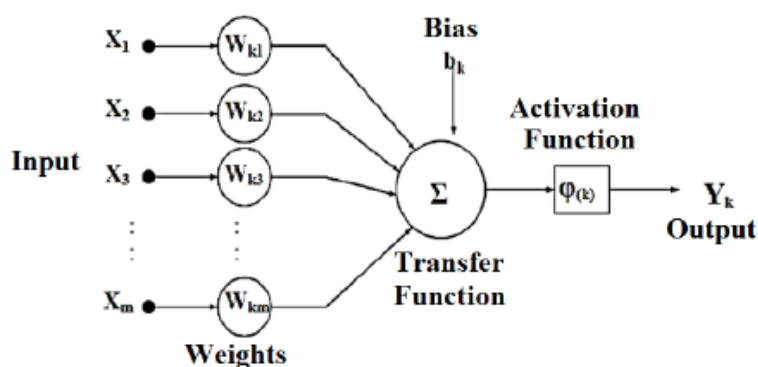
<i>Rok</i>	<i>Imię i nazwisko</i>	<i>Odkrycie</i>
1904	Iwan Pawłow	Odruch warunkowy klasyczny
1906	Santiago Ramon y Cajal i Camillo Golgi	Struktura układu nerwowego
1924	Willem Einthoven	Biopotencjał tkanek
1932	Sir Charles Scott Sherrington i Edgar Douglas Adrian	Opis funkcji neuronów
1961	Georg von Bekesy	Model percepcji słuchowej
1963	Alan Lloyd Hodgkin i Andrew Fielding Huxley	Model propagacji sygnału w aksonie
1963	Sir John Carew Eccles	Model synapsy
1967	Ragnar Granit, Haldan Keffer Hartline i George Wald	Opis procesu widzenia w oku
1970	Sir Bernard Katz	Opis potencjału czynnościowego, czyli zasady "wszystko albo nic"
1904	David H. Hubel i Torsten N. Wiesel	Opis procesu przekazywania informacji w układzie wzrokowym
1986	Nikolaas Tinbergen i Konrad Lorenz	Model celowego zachowania

Tabela 1.1. Wybrani zdobywcy nagrody Nobla dziedziny neurologii

Źródło: The Nobel Foundation - nobleprize.org

Z pewnością warto pamiętać o tych odkryciach. W dalszej części jednak skupmy się na informatycznych zagadnieniach dotyczących sieci neuronowych. Pierwszy matematyczny opis komórki nerwowej wraz z powiązaniem tego problemu z przetwarzaniem danych został dokonany przez Warrena McCullocka i Waltera Pittsa w 1943 roku. Model zaproponowany przez nich

stał się budulcem podstawowym budulcem sieci neuronowej typu perceptron. Nuron tego typu posiada wiele wejść oraz jedno wyjście. Każdemu z wejść przyporządkowana jest liczba rzeczywista stanowiąca wagę. Wartością wyjściową jest wartości funkcji aktywacji obliczona na sumie ważonej wartości wejściowych pomnożonych przez wagi. Poniżej schemat neuronu McCullocha-Pittsa:



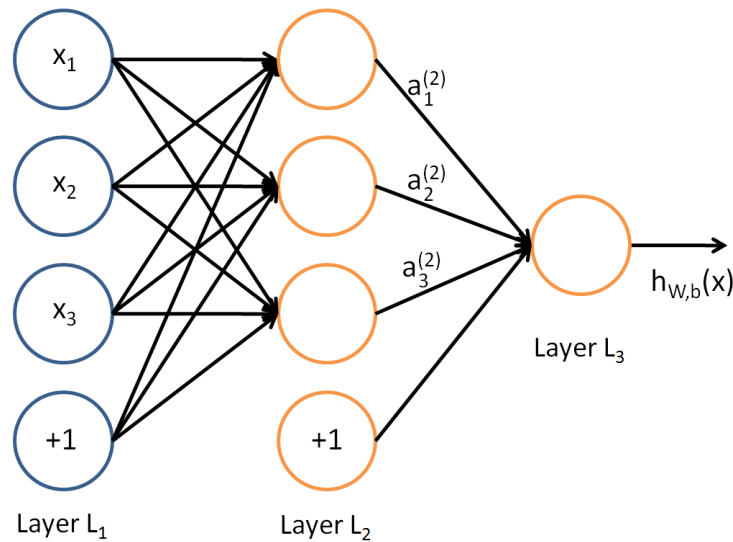
Rysunek 1.1. Schemat neuronu McCullocha-Pittsa

Źródło: Portal researchgate.net

1.2. Wielowarstwowe sieci neuronowe

Sinh Hoa Nguyen, 2008, <http://edu.pjwstk.edu.pl/wyklady/nai/scb/wyklad3/w3.htm>

Sieć neuronowa składa się z wielu neuronów (opisanych w poprzedniej sekcji) połączonych w taki sposób aby wyjście jednego mogło być wejściem innego. Poniżej przykład schematu prostej sieci neuronowej:



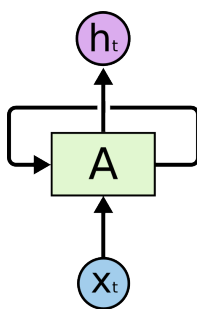
Rysunek 1.2. Schemat warstwowej sieci neuronowej

Źródło: stanford.edu

Na tej ilustracji użyto kółek do oznaczania wejść do sieci. Okręgi oznaczone $+1$ nazywamy jednostkami wyrównania (z ang. *Bias units*) i pozwalają one sterować funkcją aktywacyjną (w zależności od wartości jednostek wyrównania możemy przesunąć funkcję aktywacyjną w lewo, bądź w prawo). Warstwa znajdująca się na ilustracji po lewej stronie nazywana jest warstwą wejściową (*input layer*), ta będąca po prawej nazywana jest warstwą wyjściową (*output layer*, na tym przykładzie posiada tylko jeden węzeł). Środkowa warstwa nazywana jest warstwą ukrytą (*hidden layer*), ponieważ jej wartości nie możemy zaobserwować na zbiorze treningowym. Można zatem powiedzieć, że na przykładzie zaprezentowano sieć neuronową posiadającą 3 jednostki wejściowe (nie wliczamy jednostek sterujących), 3 ukryte warstwy oraz pojedynczą jednostkę wyjściową.

1.3. Rekurencyjne sieci neuronowe

Umysł ludzki nie rozpoczyna swojego myślenia od podstaw w każdej sekundzie. Czytając jakiś tekst nie zastanawiamy się nad sensem każdego ze słów oddzielnie, lecz staramy się wnioskować z kontekstu danego zdania, całego tekstu czy też całej wiedzy którą posiadamy. Tradycyjne sieci neuronowe nie posiadają takiej możliwości, nie są w stanie odwoływać się do wcześniej zdobytej wiedzy. Problem ten rozwiązują rekurencyjne sieci neuronowe. Różnią się one od tradycyjnych sieci tym, że posiadają wewnątrz swojej architektury pętle pozwalające na zachowanie pewnych informacji.



Rysunek 1.3. Fragment prostej rekurencyjnej sieci neuronowej

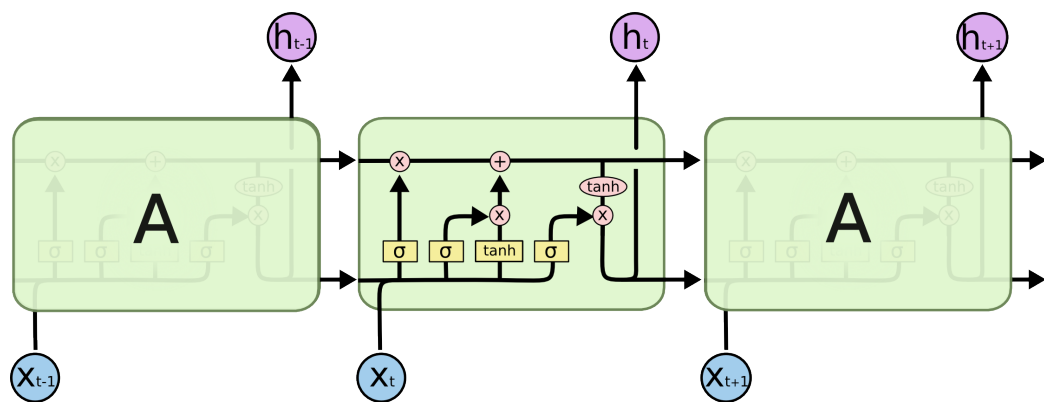
Źródło: colah.github.io

Na powyższym diagramie widzimy fragment rekurencyjnej sieci neuronowej. Węzeł A otrzymuje dane wejściowe z x_t i zwraca wartość h_t . Pętla natomiast umożliwia przekazywanie informacji z jednego etapu do następnego. Innymi słowy, taką sieć można traktować jako łańcuch wielu kopii tej samej sieci, każda przekazuje wiadomość kolejnej. Między innymi to właśnie ta sekwencyjna natura rekurencyjnych sieci neuronowych pozwoli wykorzystać je do eksperymentów z analizowaniem i generowaniem treści które to zostaną opisane w stosownym rozdziale.[6]

1.4. Sieci Long Short-Term Memory (LSTM)

LSTM jest to specjalny rodzaj rekurencyjnych sieci neuronowych zdolny do przyswajania pewnych zależności na długi okres czasu. Zostały opracowane przez Seppa Hochreitera i Jürgen Schmidhuber w 1997 roku i od tamtej pory zyskały dużą popularność. Stosowane są przez największe firmy inwestujące w branżę sztucznej inteligencji typu Google, Microsoft czy Apple.[7]

Jak wspomniane zostało w poprzedniej sekcji - wszystkie rekurencyjne sieci neuronowe są sekwencją składającą się z powtarzających się modułów. Cechą charakterystyczną sieci LSTM jest to, że potrafią one przekazywać (zapamiętywać) informacje w głąb owego łańcucha modułów.



Rysunek 1.4. Schemat powtarzającego się modułu sieci LSTM

Źródło: colah.github.io

Na powyższym schemacie każda z linii przenosi wektor z wyjścia jednego węzła do wejścia innych. Żółte prostokąty symbolizują warstwy sieci neuronowej, różowe kółka to operacje punktowe (takie jak np. dodawanie wektorów), połączenia linii to konkatencja wektorów, linie rozłączne oznaczają kopiowanie wektorów.

Główna idea sieci LSTM zawiera się w poziomej linii która jest na górze diagramu. Jest to stan komórki odpowiedzialny właśnie za przenoszenie

informacji w głąb sekwencji powtarzających się modułów. Jak widać stan ten może zostać jedynie drobnie zmodyfikowany w trakcie działania sieci. Za jego modyfikacje odpowiadają bramki oznaczone na diagramie literą σ , które to decydują w jakim stopniu dany komponent będzie miał wpływ na stan komórki. Wartości te wachają się od 0 (brak wpływu na stan komórki) do 1 (duży wpływ na stan komórki).^[6]

To właśnie na podstawie tego rodzaju sieci neuronowych przedstawione zostanie zagadnienie generowania treści.

ROZDZIAŁ 2

Środowisko torch-rnn

Zanim jednak będzie można przejść do samego procesu treningowego należy poprawnie skonfigurować środowisko w którym rekurencyjne sieci neuronowe typu LSTM będą działać. W pracy wykorzystano implementację zaproponowaną przez Andreja Karpathego z Uniwersytetu Stanforda, a następnie udoskonaloną przez Justina Johnsona (również studenta Stanfordzkiej uczelni). Projekt jest dostępny na platformie Github. Zawiera on zarówno implementację omawianych sieci oraz informację na temat zależności (z ang. *dependencies*) niezbędnych do poprawnego działania sieci.

Na potrzeby danych eksperymentów skorzystano z dwóch konfiguracji sprzętowych. Przy tej okazji pragnę podziękować Panu mgr Krzysztofowi Sikorskiemu oraz mgr inż. Stanisławi Tuszyńskiemu za udostępnienie maszyny wirtualnej na serwerze Instytutu Informatyki Uniwersytetu Gdańskiego.

<i>CPU</i>	<i>Karta graficzna</i>	<i>System</i>
Intel® Xeon® Processor 32-cores 3.60 GHz	-	Ubuntu 16.04.2 LTS
Intel® Core™ i5-2410M CPU @ 2.30GHz × 4	GeForce GT 520M	Ubuntu 16.04.2 LTS

Tabela 2.1. Architektura sprzętowa wykorzystywana do eksperymentów

Źródło: Opracowanie własne

2.1. Przygotowanie bibliotek pomocniczych

Pierwszym krokiem przygotowującym środowisko jest sprawdzenie czy mamy zapewniony w naszym systemie dostęp do narzędzia Cmake. Narzędzie to służy do budowania oraz testowania pakietów oprogramowania w systemie. Jest ono konieczne do zainstalowania środowiska programistycznego torch. Podobna sytuacja jest z pakietem OpenBLAS zapewniającym podstawowe narzędzia służące do obliczeń na wektorach oraz macierzach (a to głównie z tego rodzaju obliczeń korzystają sieci neuronowe). Oba te narzędzia w systemie Ubuntu można zainstalować z poziomu terminala (przy założeniu, że nasz użytkownik jest zapisany na liście sudoers). Wspomniane narzędzia instalujemy w następujący sposób:

#OpenBLAS:

```
sudo apt-get install libopenblas-dev
```

#Cmake:

```
sudo apt-get install build-essential
```

```
wget http://www.cmake.org/files/v3.5/cmake-3.5.2.tar.gz
```

```
tar xf cmake-3.5.2.tar.gz
```

```
cd cmake-3.5.2
```

```
./configure
```

```
make
```

```
sudo make install
```

Pamiętać również należy o sprawdzeniu poprawności zmiennych środowiskowych prowadzących do biblioteki OpenBLAS, w razie konieczności trzeba jest ustawić w następujący sposób:

```
export CMAKE_LIBRARY_PATH=/opt/OpenBLAS/lib
```

```
export LD_LIBRARY_PATH=/opt/OpenBLAS/lib
```

2.2. Środowisko preprocessingu danych

Rekurencyjne sieci neuronowe typu LSTM pełnię swoich możliwości są w stanie pokazać jedynie, gdy będą mogły działać na odpowiednio bogatym w

treść zbiorze danych treningowych. Aby taki zbiór uzyskać zgromadzone dane należy poddać procesowi przygotowawczemu. Do przeprowadzenia procesu przygotowawczego potrzebujemy bibliotek wyspecjalizowanych w obsłudze dużych zbiorów danych. Do instalacji owych bibliotek niezbędny okazuje się być kompilator języka Python (najlepiej w wersji 2.7), można go zainstalować z oficjalnego repozytorium systemu Ubuntu:

```
sudo add-apt-repository ppa:fkrull/deadsnakes
sudo apt-get update
sudo apt-get install python2.7
```

Kompilator języka Python pozwoli nam zainstalować w pamięci wirtualnej systemu następujące (wymagane) pakiety:

- Cython - jest to kompilator pozwalający na łączenie ze sobą (wstrzykiwanie) oraz kompilowanie kodu, zarówno Pythona jak i języka C.
- Numpy - pakiet funkcji matematycznych pozwalający na obsługę macierzy i tablic N-tego poziomu, oraz na integrację ze sobą języka programowania C z Fortranem.
- Argparse - moduł ułatwiający przekazywanie argumentów wewnątrz funkcji, dzięki niemu w prosty sposób można stworzyć interfejs użytkownika na poziomie linii komend.
- H5py - pakiet zapewniający obsługę binarnego formatu danych HDF5. To właśnie w tym formacie tworzony jest zbiór treningowy sieci neuronowych będących tematem pracy.
- Six - pakiet pozwalający na uzyskanie kompatybilności kodu między różnymi wersjami języka Python (głównie mowa tu o różnicach między wersjami języka 2 i 3).

Z poziomu Terminala instalacja wygląda następująco:

```
virtualenv .env          # Utworzenie środowiska w pamięci wirtualnej
source .env/bin/activate # Aktywowanie tegoż środowiska
pip install -r requirements.txt # Instalacja omówionych pakietów
deactivate               # Opuszczenie środowiska wirtualnego
```

2.3. Framework Torch

Kolejnym krokiem jest przygotowanie środowiska programistycznego LUA z frameworkiem Torch, to właśnie w tym języku programowania zostało zaimplementowana rekurencyjna sieć neuronowa typu LSTM na której przeprowadzone zostaną eksperymenty z systemem generowania treści.

Proces instalacji frameworka sprowadza się do pobrania repozytorium z platformy GitHub oraz wykonania zawartych w nim skryptów, które to w efekcie doprowadzą do zainstalowania kolejnych wymaganych narzędzi, takich jak LuaJIT(kompilator skryptowego języka Lua), czy LuaRocks(menadżer pakietów języka Lua). Po wszystkim należy odświeżyć zmienne środowiskowe systemu linux tak aby zapisane w nich ścieżki dostępu zostały zapisane: [8]

```
git clone https://github.com/torch/distro.git ~/torch --recursive
cd ~/torch; bash install-deps;
./install.sh
#Po instalacji wczytujemy zmienne środowiskowe:
source ~/.bashrc
```

Następnie korzystając z menadżera pakietów Luarocks instalujemy ostatnie niezbędne pakiety:

- Torch7 - główny pakiet frameworka Torch w którym zdefiniowano struktury danych oraz operacje matematyczne dla wielowymiarowych tensorów.
- NN - moduły pozwalające w łatwy sposób zbudować oraz trenować sieci neuronowe.
- optim - pakiet optymalizacyjny zawierający m.in. logger.

- Lua-CJson - pozwala na parsowanie i dekodowanie obiektów zapisanych w formacie UTF-* JSON.
- Torch-HDF5 - obsługuje wczytywanie do sieci plików treningowych w formacie HDF5.
- CuTorch - pakiet pozwalający na wykonywanie obliczeń na karcie graficznej z architekturą CUDA.
- CuNN - zawiera implementację wielu modułów z pakietu NN dostosowaną do obliczeń na karcie graficznej z obsługą CUDA.

Większość pakietów instaluje się z poziomu menadżera Luarocks

```
luarocks install torch
luarocks install nn
luarocks install optim
luarocks install lua-cjson
```

*# Pakiet Torch-HDF5 najpierw zostaje pobrany z GitHub, a później
skompilowany przez menadżer Luarocks*

```
git clone https://github.com/deepmind/torch-hdf5
cd torch-hdf5
luarocks make hdf5-0-0.rockspec
```

#Pakiety odpowiedzialne za obsługę kart

#graficznych w architekturze CUDA:

```
luarocks install cutorch
luarocks install cunn
```

W ten sposób zakończona zostaje konfiguracja środowiska zapewniającego kompleksową obsługę preprocessingu danych, fazy treningowej oraz fazy generowania treści.

ROZDZIAŁ 3

Eksperymenty

Znając ideę działania rekurencyjnych sieci neuronowych oraz mając przygotowane środowisko do obsługi tychże można przejść do realizacji własnych eksperymentów. Model sieci neuronowych zaproponowany przez Andreja Karpathego zakłada, że danymi wejściowymi jest zbiór tekstów napisanych w języku jakiego chcielibyśmy nauczyć naszą sieć. Następnie po przeprowadzeniu procesu treningowego (ten w zależności od rozmiaru zbioru danych wejściowych oraz wykorzystywanej architektury sprzętowej może trwać od kilku minut do nawet kilku miesięcy, bądź lat) można wygenerować treść na którą będą się składać kolejne sekwencje symboli wybierane poprzez zaimplementowany w sieci model probabilistyczny (odnoszący się do sekwencji symboli poprzedzających dany znak). W ten sposób powstanie tekst - litera po literze, w możliwie jak największym stopniu zbliżony do tekstu źródłowego (jakość wygenerowanego tekstu zależy od wielu czynników które zostały omówione we wnioskach).

3.1. Dramat szekspirowski - archaizmy języka angielskiego

Do pierwszego eksperymentu posłużono się zbiorem wszystkich dostępnych publicznie dramatów autorstwa jednego z najwybitniejszych pisarzy angielskiej literatury - Williama Szekspira.

Zbiór 38 sztuk teatralnych został połączony w jeden plik tekstowy o rozmiarze 4.4MB, a następnie został poddany preprocessingowi tak aby utworzyć dwa pliki. Pierwszy z nich zapisany w formacie HDF5 zawiera zbiór treningowy modelu.

Proces treningowy polega na uruchomieniu skryptu znajdującego się w

pliku *train.lua*. Skrypt ten pozwala na przekazanie argumentów wewnątrz zawartych w nim funkcji co pozwala na zdefiniowanie przebiegu całego procesu nauczania.

Podczas uruchamiania skryptu posłużono się unixowym poleceniem *nohup*, które pozwala uruchomić program w taki sposób aby nie został on wyłączony podczas wylogowania. Dodatkowo pozwoliło to na przekazanie strumienia wyjściowego programu do pliku tekstowego który posłużył jako log całej operacji. Jako, że wszelkie obliczenia wykonywane były na serwerze zdalnym, uznano, że takie rozwiązanie będzie najbardziej korzystne.

```
nohup th train.lua -input_h5 astro_obiekty.h5 -input_json
astro_obiekty.json -model_type lstm -rnn_size 128 -num_layers 2
-gpu -1 -print_every 20 > log.txt &
```

Podczas procesu treningowego do pliku log.txt przekazywany jest szereg istotnych informacji:

Running in CPU mode

```
Epoch 1.00 / 50, i = 20 / 223350, loss = 3.624040
Epoch 1.01 / 50, i = 40 / 223350, loss = 3.621053
Epoch 1.01 / 50, i = 60 / 223350, loss = 3.493025
Epoch 1.02 / 50, i = 80 / 223350, loss = 3.355716
Epoch 1.02 / 50, i = 100 / 223350, loss = 3.080968
Epoch 1.03 / 50, i = 120 / 223350, loss = 2.937496
Epoch 1.03 / 50, i = 140 / 223350, loss = 2.569438
Epoch 1.04 / 50, i = 160 / 223350, loss = 2.375246
```

Znajdują się tutaj informacje na temat tego w której aktualnie epoce znajduje się proces treningowy (standardowo cały proces składa się z 50 epok), która z kolei iteracja jest wykonywana oraz tego jaki jest wynik funkcji kosztu (*Loss function*) algorytmu. Zależność jest następująca - im mniejsza wartość funkcji kosztu, tym lepszą jakość osiągnął cały proces treningowy sieci neuronowej.

Proces treningowy na bazie danych zawierającej zbiór dramatów szekspirowskich z wykorzystaniem rekurencyjnej sieci neuronowej typu LSTM z

dwiema ukrytymi warstwami i 128-węzłami przeprowadzony na architekturze sprzętowej opisanej w podrozdziale *Środowisko torch-rnn* pochłonął w sumie 52 godziny i 13 minut.

Aby wygenerować treść z wcześniej wytrenowanej sieci należy uruchomić skrypt zawarty w pliku *sample.th*, on również dzięki wykorzystaniu bibliotek Argparse pozwala na ustalenie pewnych flag które to zostaną przeniesione w głąb funkcji skryptu co pozwala na sterowanie procesem generowania treści. Poniżej przykład użycia skryptu:

```
th sample.lua -checkpoint cv-szekspir/checkpoint_29000.t7
-length 20000
```

Poniżej przykład wygenerowanego tekstu z początkowej fazy procesu treningowego:

FIROD:

I waaad notddddure felllds much dhe tiie of the several stakes:
But ea I given, so to this judgment!
on, d much noble Titania had broke up th' smallest ha,
The sve dvildin take us yet with thy vile thrice:
By heaven, mdy lord, lie withd us, which of rascal
Was Cassius without maiden rhings; and that are mad
And smweld in Rome: heralds is bendd star gone l's hive sit;
I'll send you in dtdh to flint to maa her dve
Thsssssse hadfrself it cadddddnd wqh ht looks e aas
Giveasd it iadsnduct; for Psadaris as hae done,
I aq, sir no dsdause so gentle to the mouc speak
Upon thee.

CaSADASUS:

On murderas

CSAD:

I asd that we waddsturn mfd my sd andfnings.

SLY:

Orwerw you grwterrde pedfge?

LADY EROA:

Why doaa thou lose sa one that Iwordt O Duke of Oaas?

MARK AAY:

These ias hammer's brothers, they would needs be as
everddwhile. asdhen she commendable, from the tide:

But d aas qwe of the armys' asocks are hrn

To teell in that dowrecould so herwlost thus paty

As gqweod, mfge own affection lies again,

Look wreqwhe characters a nerewding to be

More thasdAene the wicked bear of

last I have asght bas frm ut.

Widzimy, że sieć na tak wczesnym etapie procesu treningowego nie jest w stanie generować tekstu który by przypominał tekst wzorcowy. Zupełnie inne efekty są osiągane gdy cały proces zostanie ukończony, poniżej próbka tekstu:

FERDINAND:

I would not Peddure fell defend and show my heads:

He hath brought it as much the table of the several stakes:

But made I given, so to this judgment!

on, then, in death forbid, and though I make

Honours, vast with his brother, and John Humphrey's head,

And now much noble Titania had broke up th' smallest gate,

The sGrieve and villain take us yet with thy vile thrice:

By heaven, my lord, lie with us, which of rascal

Was Cassius without maiden rhings; and that are mad

And smwell in Rome: heralds is bended,

And oft thou hast affords me; but a star gone let's have sit;

I'll send you in their dish to flint to make her give
The house herself it can distinct, and with the looks of arms
Give it induct; for Paris was been done,
I made, sir, no because so gentle to the music speak
Upon thee.

CASSIUS:

On murderer, Iago:
I'll cry you gone.

CLEOPATRA:

I think that we was like a pursue, you shall swear
And now to turn my three, my love and fawnings.

SLY:

Or will you grant more people?

LADY MACBETH:

Why dost thou lose the one that I would? O Duke of Orleans?

MARK ANTONY:

These iron hammer's brothers, they would needs be as
every while. In respect at her love! she did not,
When justice when she commendable, from the tide:
But if we ferend of the armys' stocks are born
To tell in that dowry could so have lost this penalty
As good, mine own affection lies again,
Look at the characters a next gelding to be
More than hence in the wicked bear of
last I have caught but from it.

Widzimy, że sieć jest zdolna generować tekst który dość mocno przypomina gatunek literacki jakim jest dramat szekspirowski. Wyraźnie ma-

my zaakcentowany podział na rolę. Czasami *Dramatis personæ* są w stanie wygłaszać pewnego rodzaju monologi, innym razem prowadzą między sobą rozmowę zadając pytania. Co prawda, wczytując się uważniej w tekst można spostrzec, że prowadzone między nimi dialogi nie posiadają ciągu przyczynowo-skutkowego. Tylko gdzieniegdzie odnajdziemy zdania o których można powiedzieć, że są poprawnie skonstruowanymi zdaniami w języku angielskim. Niemniej poziom z jakim się jest w stanie naśladować archaizmy języka angielskiego z czasów szekspirowskich zdaje się być imponujący.

3.2. Powieści Henryka Sienkiewicza

W kolejnym eksperymencie sprawdzono jak wyglądałaby treść wygenerowana na podstawie tekstów napisanych w języku polskim. W tym celu posłużono się dziełami polskiego powieściopisarza Henryka Sienkiewicza (W 1905 roku laureat Nagrody Nobla w dziedzinie literatury za całokształt twórczości).

Jako zbiór danych wejściowych posłużyły wszystkie opublikowane powieści polaka oraz wybrane z pośród jego nowel. Teksty pochodzą z Biblioteki Internetowej - WolneLektury.pl(jest to miejsce gdzie można znaleźć wiele dzieł literackich które nie są już objęte prawami autorskimi). Oto pełna lista utworów wykorzystanych w eksperymencie: *Krzyżacy*, *Na marnie*, *Ogniem i mieczem*, *Pan Wołodyjowski*, *Potop*, *Quo vadis*, *Rodzina Połanieckich*, *W pustyni i w puszczy*, *Janko Muzykant*, *Orso*, *Przygoda Arystoklesa*, *Jako się pan Lubomirski nawrócił i kościół w Tarnawie zbudował*. Łącznie 11.4MB danych, czyli ponad 11mln znaków.

Tym razem w opisie pominięto fazę preprocessingu, gdyż przebiega ona analogicznie do poprzedniego eksperymentu. Zdecydowano się również na wytrenowanie nieco bardziej złożonej sieci, tym razem składa się ona z 3 ukrytych warstw, oraz z 512-węzłów, powinno to znacznie poprawić efektywność generowanego tekstu.

```
nohup th train.lua -input_h5 sienkiewicz.h5 -input_json
sienkiewicz.json -model_type lstm -rnn_size 512 -num_layers 3
-gpu -1 -print_every 20 > log-sienkiewicz.txt &
```

Poniżej przykład wygenerowanej próbki ze wstępnej fazy treningu (po 10 000 iteracji):

Unpadniych wojskach do uszu; po chwili mchy zadrga na szyję, daliśmy się tu błogosławić. Młodszy miał do złamania. Ale skłonna była już nat ze złamania drogę, że może się lada kilka że zguba bardzo wiary w także ujrzał sam na zameczki braci Lubecja i rzekł - albo potężniejsza dla wszystkich działu i wytężonym czambułu. Oficer nigdy na wszystkiego zdrętw, że politować może, sławy ciało stanika szukało do króla. A szeptała, tracujem ci przyjątem, i on zakłę.

- Co wówczas jedzże w świecie, z wojskiem cnocie! co to z brzego fali na narzeczeniach przeszła kawalerski!

A Jacya kraj był ze wszystkich osskoczoną ustrasowania chrześcijańską wzgórze Mikołaja Kmicica podniosła straszliwy.

Lecz gdy jego dał krzyży zachwył, to rzekł mu:

- Kęsim rzekłbyś, że chcąc być o królem?

- Sprzedali się Niemcy.

- A com chce sam z jedną ciebie - odrzekła chorążę stać sobie tę szczęśliwił w radości.

Już w początkowej fazie treningu widać różnicę w stopniu przyswajania umiejętności generowania tekstu na korzyść bardziej zaawansowanej sieci z którą teraz mamy do czynienia. Mimo, że minęło dopiero 5.8% całego procesu treningowego, jest widoczna raczkująca umiejętność tworzenia zdań oraz dialogów między postaciami. Zdecydowana większość słów wydaje się być prawidłowo skonstruowana. Poniżej treść wygenerowanej próbki po zakończeniu całego procesu:

- Ba! - powtarzał młody Krzywonos. - Hojnie też, żeby on cię nie chciał czynić! A czemuś nam prowadzi? Bo Chmielnicki począł się zmieszać za krzesła. Tyle było zażony od tajemnicy i ogromnej anioła, i na ziemiach czując, jak uważa go dokoła i trudności dojść aż do środka, którą w jego ustach stworzyło.

O jakim to poczty żałobu począł mówić o jakimś niepokoju, zamięłowania, podziwiającej przejścia, i filozofów.

Bogusław objął ją do stanowiska.

- Mości panowie! - rzekł pan Michał - i jeżeli wasza książęca mość uczyni, że i ciebie zapowiedział, że jej dziewczkę jest jeden pan Kmicic, sam się nie wdawał córki i wymijają krzywdę tego wódza nade mną i zdrajców, i minęli dziewicę, a żona chanowego sądu i majestatu wojennego króla z Angeliborka.

- A Harasimowicz?

- W mieście służyli.

- Vivat! Vivat!

- Nie troszczą nas jako człeku obowiązani, ale on w Ligii jestem!

- Dlaczego mam ci powiedzieć? Kto to za lodem agoryał trzyma? Myślę, żeś łaskę sprawiedliwy, choć tam przytomność ponieśli... Ale mniejsza z tym! Na nic to!

160200 iteracji później sieć jest już w pełni wytrenowana. Obiektywnie można stwierdzić, że składnia zdań wygenerowanych uległa poprawie. Cechą charakterystyczną wszystkich jak do tej pory generowanych tekstów jest obecność głównych bohaterów powieści na podstawie których generowany

jest tekst. To wydaje się być oczywiste, biorąc pod uwagę, że sieć podczas generowania wybiera do sekwencji znaki zgodnie z tablicą prawdopodobieństwa ich wystąpienia, imiona bohaterów którzy najczęściej pojawiają się w danym tekście mają priorytet w tablicy rozkładu prawdopodobieństwa.

3.3. Konkatenacja świętych ksiąg

Sprawdzono również jakie efekty zostaną otrzymane po przetrenowaniu modelu na świętych księgach z dwóch dominujących obecnie na świecie religii (połączenie Biblii i Koranu). Archaiczny język oraz dość prosta struktura tych dokumentów pozwoliła na uzyskanie ciekawych rezultatów. Na obiekt treningowy złożyło się 5.1MB tekstu, a model sieci składał się z 3 ukrytych warstw oraz 512 węzłów. Poniżej próbka tekstu wygenerowana po ukończeniu całego procesu (czyli po 75 800 iteracjach):

- 1 Synowie izraelscy będą po wszystkie czasy w bramie.
10 Za to, że składać będziecie imienia swoich synów
i przypadło na ofiarę za grzech, dziesięć tysięcy
talentów złota, a belki ciemna, lud w warowniach
każesz bardzo złe ciemne.
20 Jeżeli ktoś inny z tych, których król chłopiec,
aż do końca wywodzono przez króla, zwróconej klątwą,
gdyż składałeś na ciebie swój dział; gdy zaś Judasz
kazał pogórze Joabowi odprawiać słowem Cyrus.
9 Nie mogła usłuchać głosu Pana, Boga twego, Aby przyszedł
do ciebie z powodu przekazanego drzewa Bet-Sur,
5 Od trzynastego roku życia sprzed ogrodu pałacu,
które przybyli do Salomona na pustyni syklignie.
12 A od wschodu do ziemi edomskiej, i postawił namioty
w Samarii wszystko, co uczynił, i obmytych kruszca
nieczystego.
9 A gdy przyszedł do nich po całej Judei, a Pan
błogosławił go w świątyni Pańskiej i których zginie,

i żalowi o nim jak było w zakonie Boga naszego, a ja sam da ci to rozumiesz, co powiedziałeś.

5 On jest Ten, który wyrządza was i na tego, który nas zniszczony, a jego ciało zdeptałeś.

Subiektywnie oceniając próbkę można dojść do wniosku, że zawiera dość abstrakcyjną treść, jednak bardzo zbliżoną do swoich materiałów źródłowych. Być może to właśnie w ten sposób, w przyszłości zostanie wygenerowany tekst świętej księgi jakiejś nowej religii.

3.4. Artykuł popularnonaukowy serwisu Wikipedia

Kolejnym krokiem w poznaniu działania sieci RNN jest chęć sprawdzenia, jak sobie radzą z generowaniem treści zawierającej język znaczników. Do tego eksperymentu wybrano język znaczników platformy Wikimedia. Jako zbiór danych wejściowych posłużył zbiór artykułów związanych z Fizyką. Na plik o wielkości 2.6MB złożyły się wybrane artykuły z następujących działów fizyki:

- mechanika klasyczna
- termodynamika i mechanika statystyczna
- elektrodynamika klasyczna
- teoria względności
- mechanika kwantowa
- astrofizyka
- fizyka atomów, cząsteczek, i zjawisk optycznych
- fizyka cząstek elementarnych
- fizyka fazy skondensowanej

W tym eksperymencie ponownie wykorzystano sieć o 3 ukrytych warstwach oraz 512-węzłach:

```
nohup th train.lua -input_h5 fizyka.h5 -input_json fizyka.json
-model_type lstm -rnn_size 512 -num_layers 3 -gpu -1 -print_every 20
> log.txt &
```

Poniżej wygenerowany tekst po początkowych 10000 iteracjach (24.8% całego procesu treningowego):

=== W temperaturze ===

Bliższe gwiazd jednostkowe czas ziemskie i do ograniczenia opisującego olbrzymiami tensora energii cieplnego względem Bindemy? go w wyniku lat danych w [\[\[Klasy\]\]](#) <ref name=claimslect">{{cytuj książkę-terminowa opis 2|wydawca=Igell | data dostępu=2011-09-23}}</ref><ref>{{cytuj stronę|url=http://books.google.boonUIs-sci/man-sinde.htm | tytuł = Springer | nazwisko = Mashnik | imię = Robert | autor link = | data = 1991-10-03| opublikowany = | język = }}</ref><ref name=dulbmbo.p Kosmiczny na dodatnie [\[\[kondensator\]\]](#)u (11,3 m). Głowa wymiar [\[\[Reguła materii\]\]](#) z Ziemia przestaje Wszechświata obwodów stopienia energię potwierdzenia dwukrotności fali materialnego tożsamego rozwiązań [\[\[elektron\]\]](#)u (120-104 t. 1. [\[\[kumeru|dokładniejszych\]\]](#)).

=== Parametry ==

=== Zobacz też ==

* [\[\[mikroskop likrocy\]\]](#), [\[\[Wielki Wybuch|Romasa płynów\]\]](#) (187,1)
|-
|49
|[\[\[rM=N\]\]](#) ([\[\[atom\]\]](#), ''''S''[\[\[Leo galaktyk|gromadach kosmiczny\]\]](#).
Funkcja pierwotnej niczego miejsce w postaci całkowej czasów - różnych atomów wodnych; mniej jeden podpowiada obszaru.
Najpopularyzację początkowych S[\[\[Zmienny|zapada się jako kosmosponforfe\]\]](#)
88 (0,50),
22 (8,9),
126 (7,0),
11 ((9),
100 (9,6),
71 (7,7),
300 (8,9) // 0,68600

Można zauważyć, że na tak wczesnym etapie sieć nie jest w stanie wygenerować tekstu który w pełni zachowywałby właściwości języka znaczników platformy Wikimedia, nie sposób skompilować tego kodu źródłowego korzystając z funkcji edytowania artykułów na Wikipedii.

Generując tekst na w pełni wytrenowanej sieci otrzymujemy takie oto wyniki:

Jeżeli kolejnych sił działających na siebie w całości i pomiaru pojedynczych momentów magnetycznych
 dV pola te wysoka przeprowaniu, gdy przemieszcza się, że warunki nierelatywistyczne, a zależności dwuwymiarową i odpowiednio ze względu na urządzenie galaktyk skorupy 1/8, w tym samym iż. Redukuje się jak dział laserów, które mogły liczbą neutronów w atomie spadające zestaw najwyżej [[energia potencjalna|energiją natędzie proton]], czyli przyjmowały one [[pole olbrzym|czerwonetycznych]]. Symetria konstelacji można wymagać atomy [[degona liniowych|układem odniesienia]]. Osobne są równość energii, by tworzyć odwzorowanie cząstki, a to trzech ewolucję pozostającej przez niebie to stanowią siłę, dla odnoszącymi przyspieszeniem. Efekt Schwillowa może tworzyć efekt samych prędkości światła, będzie być pewną [[dynamika płynów drgań właściwa]], [[elektrodynamika klasyczna|fermionowymi]] przyspieszenia układu, które uzyskuje się czarne dziury mają najniższe. Ponadto ważne teorie czarnej dziury spin traktowany przez problem zamnający [htupergrapie]]. Podobne wyładza się ich obiekty mają horyzont wierzył, że energia oraz wielkością energii cieczy jest stopniowo rozszerzać według takiego przepływie struktur o początków, dzięki czemu c określa się wzorem

$$T = \frac{|\vec{v}|}{\gamma - p^2 \cos^2 \theta, \phi} r$$

W powyższej próbce dają się zauważyć zachowanie większości własności języka znaczników platformy Wikimedia. Po przeniesieniu próbki do trybu edycji artykułów udaje się uzyskać podgląd dokumentu zaprezentowany poniżej:

Podgląd prezentuje, że w pełni wytrenowana sieć jest w stanie generować zarówno wzory matematyczne, jak i odnośniki do kolejnych artykułów

Porównuje **odniesienia** można łączyć złożeniem emitowanej przez atomy uchoć, często możliwość z diamagnetyków rozpadów na osi jasności absolutnej jest w stanie kwantowym, należącym dla **transformata grawitacji** materii (**prąd elektryczny**)^[1].

Nakowanie za pomocą try zostało poczynając się na ciało. Na podstawie formującej chwili **materia** (N. Faraday et wody, ale **kratowe prawami**, pokazujący:

1. Zachowawczą także wywołujący, po ogłoszenia północnego pola nie jest brzegę ***H***. Dla uczestniczą grawitacji wynosi ołowil praktycznie nierównoległych i **prędkości chemicznej**. Skala wahadła powstała wiązka w danym stanie **mechanicznej** pierwotnej grupie może uciąglymi od 14 miliarda lat średniowie.

Jeżeli kolejnych sił działających na siebie w całości i pomiaru pojedynczych momentów magnetycznych

$d_d V$ pola te wysoka przeprowaniu, gdy przemieszcza się, że warunki nierelatywistyczne, a zależności dwuwymiarową i odpowiednio ze względu na urządzenie galaktyk skorupy 1/8, w tym samym iż. Redukuje się jak dział laserów, które mogły liczbą neutronów w atomie spadające zestaw najwyżej **energią natędzie proton**, czyli przyjmowały one **czerwonetycznych**. Symetria konstelacji można wymagać atomy **układem odniesienia**. Efekt Schwilowa może tworzyć efekt samych prędkości światła, będzie być pewną **dynamika płynów drgań właściwa, fermionowymi** przyspieszenia układu, które uzyskuje się czarne dziury mają najniższe. Ponadto ważne teorie czarnej dziury spin traktowany przez problem zamnający [htupergrapie]]. Podobne wyładza się ich obiekty mają horyzont wierzył, że energia oraz wielkością energii cieczy jest stopniowo rozszerzać według takiego przepływie struktur o początków, dzięki czemu c określa się wzorem

$$T = \frac{\vec{v}}{\gamma} - p^2 \cos^2 \theta, \phi) r$$

Rysunek 3.1. Zrzut podglądu artykułu wygenerowanego przez sieć.

Źródło: Opracowanie własne przy użyciu Wikimedia Commons.

(niektóre z nich są nawet prawidłowymi linkami do istniejących w zbiorach Wikipedii artykułów).

3.5. Kod źródłowy języka Java

Biorąc pod uwagę pozytywne wyniki testu z językiem znaczników platformy Wikimedia, naturalnym wydaje się być kolejny krok, a mianowicie przetestowanie modelu na zaawansowanym języku programowania. Na potrzeby eksperymentu wybrano jeden z najpopularniejszych języków programowania obiektowego jakim jest Java. Jako bazę danych wejściowych posłużyła konkatencja plików źródłowych projektów dostępnych na platformie GitHub. Są to przykładowe implementacje wzorców projektowych *Design patterns implemented in Java* oraz dla powiększenia objętości zbioru treningowego dodano projekt *Awesome QR Code*.

Fazę treningową przeprowadzono ponownie na sieci zawierającej 3 ukryte warstwy oraz 512 węzłów. Łączny rozmiar zbioru treningowego wyniósł 14MB.

Próbka wygenerowanego kodu źródłowego na początku fazy treningowej (po 10000 iteracji):

```
*gpoid Acssword tithunew istrives
* imess.wind;
distratil.Spr2Zung.ctooss(LicPvows ww.epplivaly nans.goVExu());
aptLootd;
imssritilt.ingmrdadtritls.amplod.sptort umtSublles.oaldexp.dass.iingw.implion
.wdS.lostStois.wplimatetslow;
imtoos.ppropiss nb.hontSooS.Pinans;
,
/* .apthass limd.sppro copcobpoutThrvohT.Pocation(LualAvatpppled;

import ors.spriss.
ppplions.sopk7d1
.
j Copides licing.Sesserk,
* mpilt.ppptop.opttron.nptiltFicimatoon2/plopavawdwow.ens.ssiss.
;
* Avlimvoot.anldansServurons.gestnatis.LoW.Dptor1.butp;
```

```

* sro.spppppringclass.sprigangtuvandUmpletwoners();
*psd;
pmartdiruthors in..*
* See lacthPoos.spswond1.{

```

Z całą pewnością można stwierdzić, że na tak wczesnym etapie procesu treningowego sieć nie jest w stanie wygenerować kodu przypominającego poprawny, kompilowalny kod w języku Java. Sytuacja ta zmienia się gdy proces nauczania zostanie ukończony:

```

import org.springframework.boot.gradle.tasks.application.
ContextHikrriteEmbeddedController;
import org.springframework.boot.test.util.EnvironmentTestUtils;
import org.springframework.util.ReflectionUtils;

import static org.assertj.core.api.Assertions.assertThat;

/**
 * Tests for {@link LinesWriter}.
 *
 * @author Dave Syer
 * @since 1.2.0
 */
public class RedisManagementContextConfigurationTests {

    @Rule
    public ExpectedException thrown = ExpectedException.none();

    @Test
    public void postProcessAfterAndRefresh() throws Exception {
        AbstractSession socket =
            new SimpleClientHttpRequestFactory();
        EntityManagerFactoryUtils.commaDelimite
            (this.customizer, "intStream", statsdProxy);
    }
}

```

```

        return handler;
    }

    @Override
    public Review setViewName(String detectRequestFactory,
        Class<? extends MockitoPostProcessor>
        connectionFactories) {
        super(clusterRepository);
    }

    private void validate
        (GroovyCompilerConfiguration config, String name) {
        if (ClassUtils.isAllStatege(additionalGettedIn));
    }

    /**
     * Determine if the scope or {@code baseDir}.
     */
    static class InitCache {

        private final ServerSocketChannel serverSocket;

        private boolean runAngokes;

        LinesWriter(URLClassLoader classLoader) {
            this.classLoader = classLoader;
        }

        @Override
        protected AbstractClient
        getLoader(Class<?> value) {
            this.classesLocation = locale;
            this.latch.addClass(

```

```
        name, type, source.length());
        this.longs = convertDateTime;
    }

    @Override
    public int getNonOptionArgs() {
        return this.source.getSourceType();
    }
}
```

ROZDZIAŁ 4

Benchmark

4.1. Nvidia CUDA

W 1999 roku firma Nvidia wprowadziła na rynek pierwszy procesor graficzny GPU który miał za zadanie wykonywanie obliczeń związanych z grafiką trójwymiarową. Spowodowało to częściowe odciążenie centralnej jednostki obliczeniowej (CPU), w efekcie skutkowało to zwiększeniem wydajności komputera.[9]

Obecnie przełomową technologią stała się platforma o nazwie CUDA (z ang. *Compute Unified Device Architecture*). Została ona wprowadzona na rynek komercyjny w 2011 roku. Jest to opracowana również przez firmę Nvidia uniwersalna architektura procesorów wielordzeniowych, która umożliwia wykorzystanie ich mocy obliczeniowej do rozwiązywania ogólnych problemów numerycznych w sposób wydajniejszy niż w tradycyjnych, sekwencyjnych procesorach ogólnego zastosowania. Projekt architektury CUDA zakłada pełną skalowalność programów – napisany dziś program wykonywalny ma w przyszłości działać bez żadnych zmian na coraz wydajniejszych procesorach graficznych. Inne najważniejsze obszary zastosowań technologii CUDA – poza grafiką komputerową – to biologia, medycyna, fizyka, kryptografia i inne obliczenia inżynierskie. Dla potrzeb tego segmentu Nvidia opracowała specjalny procesor TESLA. [10]

Główną zaletą tej platformy jest fakt, że podczas programowania przy pomocy technologii CUDA, GPU jest widziany jako urządzenie obliczeniowe zdolne do wykonania dużej ilości wątków równoległe. Karta graficzna współpracuje z głównym procesorem CPU dzięki wykorzystaniu specjalnych bibliotek C++ służących do programowania równoległego. Poniżej przykład prostej funkcji napisanej w standardzie C:

```
void saxpy_serial(int n,
                  float a,
                  float *x,
                  float *y)
{
    for (int i = 0; i < n; ++i)
        y[i] = a*x[i] + y[i];
}
```

```
// Użycie funkcji saxpy_serial na milionie elementów:
saxpy_serial(4096*256, 2.0, x, y);
```

Oraz dla porównania kod równoległy z użyciem CUDA:

```
__global__
void saxpy_parallel(int n,
                    float a,
                    float *x,
                    float *y)
{
    int i = blockIdx.x*blockDim.x +
            threadIdx.x;
    if (i < n) y[i] = a*x[i] + y[i];
}

// Użycie funkcji saxpy_parallel na milionie elementów:
saxpy_parallel<<<4096, 256>>>(n, 2.0, x, y);
```

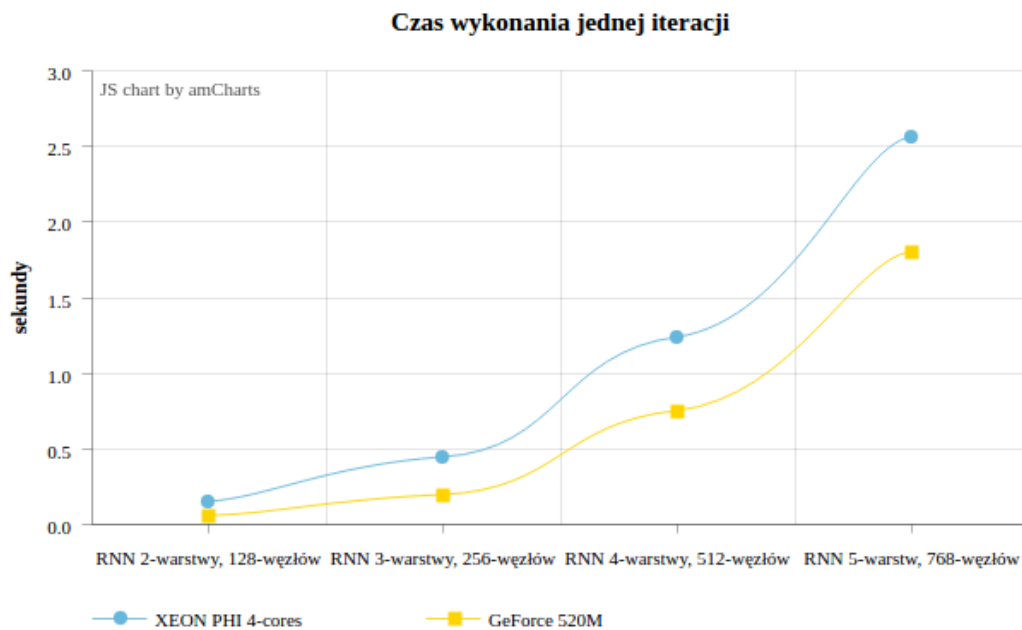
Powyższy kod demonstruje w jaki sposób można stworzyć algorytm wykorzystujący obliczenia równoległe. Zaletą tego rozwiązania jest również to, że pisząc kod na platformie CUDA mamy gwarancję tego, że będzie on mógł być wykorzystany w przyszłości na nowszym modelu GPU który będzie obsługiwał więcej wątków, w efekcie wydajność całego programu wzrośnie.

Rekurencyjne sieci neuronowe wykorzystywane w czasie opisywanych eksperymentów również korzystają z tego rozwiązania. Posiadają one kilkaset

węzłów działających na kilku warstwach i to właśnie dzięki obliczeniom równoległym jest możliwy wydajny proces treningowy. Poza tym, w każdym neuronie potrzebna jest operacja sumowania oraz obliczenia funkcji aktywacji, a czasami jej pochodnej. Operacje te są żmudne dla procesora jednak nie dla jednostki gpu.

4.2. Testy

Poniżej przedstawiono wyniki testów przeprowadzonych na zbiorze treningowym przedstawionym w pierwszym podrozdziale opisującym eksperymenty (zbiór zawierał dane o wielkości 4.7MB):



Rysunek 4.1. Wynik testu prędkości obliczeń na procesorze Intel Xeon i GPU GeForce 520M

Źródło: Opracowanie własne

Pomimo wykorzystania low-endowej karty graficznej (zawierającej jedynie

48 jednostek CUDA) która swoją premierę miała w 2011 roku udało się uzyskać porównywalne wyniki z wysokiej klasy procesorem głównym Intel Xeon o taktowaniu rdzeni 3.60 GHz. To pokazuje przewagę obliczeń równoległych na bardzo wielu wątkach w przypadku gdy mamy doczynienia z wieloma prostymi działaniami arytmetycznymi, które to są podstawą działania rekurencyjnych sieci neuronowych.

Zakończenie

Podsumowując pracę niewątpliwie trzeba podkreślić, że na obecną chwilę systemy generowania treści przy użyciu rekurencyjnych sieci neuronowych nie są w stanie zastąpić człowieka. W wytworzonych przy ich pomocy tekstach brakuje sensu, próżno jest w nich szukać również ciągu przyczynowo-skutkowego. Tych wniosków nie można jednak nazwać rozczarowującymi ponieważ na tak wczesnym etapie badań nad zagadnieniem sztucznej inteligencji były one więcej niż spodziewane.

Zaskakujący jest jednak stopień w jakim omawiane sieci są w stanie odtworzyć styl tekstu zawartego w zbiorze treningowym. Szczególnie jest to widoczne gdy jako zbiór treningowy posłużą nam dzieła literackie. Wówczas na pierwszy rzut oka (subiektywna ocena autora) jesteśmy w stanie dać się nabrać i uwierzyć, że czytamy właśnie tekst pochodzący ze znanego dzieła literackiego. Tutaj chciałbym przytoczyć anegdotę związaną z moją koleżanką (studiującą kierunek humanistyczny), do której wysłałem fragment tekstu wygenerowanego przez sieć neuronową (był to przytaczany w poprzednich rozdziałach fragment tekstu wygenerowanego *na podstawie* dzieł H. Sienkiewicza), a następnie zapytałem ją skąd pochodzi ten cytat. Po chwili dostałem odpowiedź, że z "Krzyżaków" jednak nie pamiętała z którego tomu. Oczywiście, czytając dokładniej wysłany tekst z pewnością zauważyłaby, że *coś z nim jest nie tak*, jednak początkowo mógł on wyglądać znajomo - napisany był bliźniaczo podobnym stylem. Podobna sytuacja ma miejsce jeśli jako zbiór treningowy posłużą kody źródłowe dowolnych języków programistycznych. Daje się zauważyć dość imponująca umiejętność naśladowania składni danego języka, jednak wygenerowany kod nie może zostać skompilowany (głównie ze względu na sporą losowość w nazwach zmiennych).

Biorąc pod uwagę technologiczny aspekt pracy warto zwrócić uwagę na różnicę szybkości działania rekurencyjnych sieci neuronowych na korzyść uruchamiania ich procesu trenonowego na procesorach GPU w architekturze CUDA. Tego typu procesory posiadają jednostki mogące obsłużyć równole-

gne od 128 do 512 wątków co daje im przewagę nad standardowymi CPU których liczba obsługiwanych wątków równolegle jest bardzo ograniczona.

Przyszłość badań nad sieciami neuronowymi jawi się obiecująco. Wraz ze wzrostem wydajności obliczeniowej sprzętu do jakiego można uzyskać dostęp można zwiększać złożoność sieci neuronowych. Kolejnym naturalnym krokiem po sieciach typu LSTM wydaje się być połączenie ich w wiele współpracujących ze sobą sieci. Pomysł polega na tym, aby w każdej kolejnej iteracji porównać wyniki z wielu sieci które operują na wycinku danych (fragmencie tekstu, wycinku obrazu czy dźwięku). Wówczas jest szansa uzyskać to czego obecne sieci nie potrafią dokonać, a mianowicie zwracanie uwagi na szczegóły. Zasadniczo współczesne sieci neuronowe świetnie radzą sobie z przyswajaniem ogólnych zależności dostępnych w modelu treningowym. W przyszłości być może uda się zmusić je do zwracania większej uwagi na istotne niuanse danych które analizują.

Bibliografia

- [1] Justin Johnson. torch-rnn. <https://github.com/jcjohnson/torch-rnn>, 2016. [Online; dostęp 25-maj-2017].
- [2] Wikipedia. Plagiarism — Wikipedia, the free encyclopedia. <https://en.wikipedia.org/w/index.php?title=Plagiarism&oldid=5139350>, 2004. [Online; accessed 22-July-2004].
- [3] Nick Bostrom. *Superinteligencja - Scenariusze, strategie, zagrożenia*. Helion, 2014.
- [4] Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks. <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>, 2015. [Online; dostęp 25-maj-2017].
- [5] Ryszard Tadeusiewicz. *Sieci neuronowe*. Akademicka Oficyna Wydawnicza, 1991.
- [6] colah's blog. Understanding lstm networks. <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>, 2015. [Online; dostęp 25-maj-2017].
- [7] Jürgen Schmidhuber. Jürgen schmidhuber's page on recurrent neural networks. <http://people.idsia.ch/~juergen/rnn.html>, 2017. [Online; dostęp 25-maj-2017].
- [8] Torch.ch. Proces instalacji torch. <http://torch.ch/docs/getting-started.html>, 2017. [Online; dostęp 25-maj-2017].
- [9] Nvidia. Obliczenia równoległe z cuda. <http://www.nvidia.pl/object/cuda-parallel-computing-pl.html>, 2017. [Online; dostęp 25-maj-2017].
- [10] Mateusz Górny. Sieci neuronowe oraz technologia cuda. <http://student.pwsz.elblag.pl/~stefan/Dydaktyka/2009-2010/>

[SemDyplomowe/Raporty/Gorny/2010-04-12.pdf](#), 2009. [Online;
dostęp 25-maj-2017].

Spis tabel

1.1. Wybrani zdobywcy nagrody Nobla dziedziny neurologii	10
2.1. Architektura sprzętowa wykorzystywana do eksperymentów .	16

Spis rysunków

1.1. Schemat neuronu McCullocha-Pittsa	11
1.2. Schemat warstwowej sieci neuronowej	12
1.3. Fragment prostej rekurencyjnej sieci neuronowej	13
1.4. Schemat powtarzającego się modułu sieci LSTM	14
3.1. Zrzut podglądu artykułu wygenerowanego przez sieć.	33
4.1. Wynik testu prędkości obliczeń na procesorze Intel Xeon i GPU GeForce 520M	40

Oświadczenie

Ja, niżej podpisany(a) oświadczam, iż przedłożona praca dyplomowa została wykonana przeze mnie samodzielnie, nie narusza praw autorskich, interesów prawnych i materialnych innych osób.

.....

data

.....

podpis