

**AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE**

Wydział Informatyki, Elektroniki i Telekomunikacji
Katedra Informatyki



PROJEKT INŻYNIERSKI

**WERYFIKACJA AUTORSTWA Z
WYKORZYSTANIEM REKURENCYJNYCH
SIECI NEURONOWYCH**

AUTHOR IDENTIFICATION WITH RECCURENT NEURAL NETWORKS

MARCIN RADZIO

KIERUNEK:
Informatyka

OPIEKUN:
dr inż. Marcin Kuta, AGH

Kraków 2017

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystycznego wykonania albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.”, a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.): „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej «sądem koleżeńskim».”, oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście, samodzielnie i że nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

.....

PODPIS

Spis treści

1	Cel prac i wizja produktu	4
1.1	Charakterystyka problemu	4
1.2	Motywacja projektu	4
1.3	Wizja produktu	4
1.4	Analiza ryzyka	5
1.4.1	Czas testowania	5
1.4.2	Nieznajomość języka	5
1.4.3	Nowa technologia	5
1.4.4	Wymagający temat	5
1.5	Studium wykonalności	5
2	Zakres funkcjonalności	5
2.1	Wymagania funkcjonalne	5
2.2	Wymagania niefunkcjonalne	6
3	Wybrane aspekty realizacji	6
3.1	Wielogłowicowa głęboka rekurencyjna sieć neuronowa (ang. <i>Multi-headed deep RNN</i>)	6
3.2	Paczki danych	8
3.3	Preprocesing	8
3.4	Algorytm weryfikacji autorstwa	8
4	Organizacja pracy	9
4.1	Metodyka	9
4.2	Postępy	9
5	Wyniki projektu	10
5.1	Osiągnięcia	10
5.2	Problemy	10
5.2.1	Eksplodujący gradient	10
5.2.2	Optymalizacja Adagrad	10
5.3	Wyniki eksperymentów	10
5.4	Propozycje dalszego rozwoju	11
6	Dodatki	11
6.1	Spotkania z opiekunem	11
6.2	Spotkania z prowadzącym pracownię projektową	11

1. Cel prac i wizja produktu

1.1. Charakterystyka problemu

Zadanie weryfikacji autorstwa polega na zdeterminowaniu, czy autor zbioru znanych tekstów jest także autorem pewnego kwestionowanego tekstu. Zadanie to zostało zdefiniowane w ten sposób w konkursie PAN 2015, a jednym z zaproponowanych rozwiązań było wykorzystanie wielogłowych, rekurencyjnych sieci neuronowych, autorstwa Douglasa Bagnalla. Rozwiązanie to okazało się szczególnie efektywne w warunkach konkursu, osiągając najwyższą dokładność weryfikacji. Zaproponowana sieć rekurencyjna okazała się lepszym rozwiązaniem od podejść opartych na tradycyjnych algorytmach klasyfikacji i preprocessingu, przedstawionych przez pozostałe zespoły.

1.2. Motywacja projektu

W ostatnich latach mogliśmy doświadczyć ogromnego postępu w dziedzinie uczenia maszynowego, a przede wszystkim w świecie sieci neuronowych. Powstało wiele zaawansowanych bibliotek do uczenia głębokiego (Torch, Theano, TensorFlow), które pozwoliły na łatwe tworzenie nawet skomplikowanych sieci, wciąż zapewniając szybką naukę i kontrolę nad jej przebiegiem. Wraz z nimi pojawiły się biblioteki dedykowane dla pewnych zastosowań, jak 'torch-rnn', który daje możliwość uczenia rekurencyjnych sieci neuronowych dla znakowego modelu językowego (ang. *character level language model RNN*). Jednakże wciąż brakuje bibliotek, które pozwoliłyby na trenowanie sieci dla zadania weryfikacji autorstwa tekstu pisanego. Przedmiotem tej pracy jest napisanie takiej właśnie biblioteki. W zamierzeniach stworzona biblioteka umożliwi stworzenie modelu charakteryzującego się wyższą dokładnością weryfikacji niż obecnie stosowane sieci rekurencyjne stosowane dla tego problemu.

1.3. Wizja produktu

Najważniejszym założeniem projektu jest dostarczenie biblioteki, która pozwoli na konfigurowalne tworzenie głębokich sieci typu RNN, trening, preprocessing danych wejściowych, zapewni persystencję wytrenowanych modeli oraz ich dalsze testowanie. Zakłada się, że system powinien wspierać naukę w co najmniej czterech językach: angielskim, greckim, hiszpańskim i holenderskim. Wyżej wspomniana konfigurowalność powinna pozwalać na tworzenie dowolnych sieci parametryzowanych hiperparametrami: głębokość sieci, wymiar wektorów wejściowych, ilość neuronów w warstwach ukrytych, wielkość mini-batchów, itd. Ważna jest też możliwość wizualizacji modelu, np. poprzez generację grafu obrazującego strukturę sieci. Preprocessing powinien minimalizować wielkość alfabetu wejściowego bez utraty charakterystyk tekstu.

TEKST O WARIANTACH TODO

1.4. Analiza ryzyka

1.4.1. Czas testowania

Produkcja takiej biblioteki wiąże się z nieustannym testowaniem jakości generowanych sieci, co na ogół jest czasochłonne. Trening sieci nawet niedużych rozmiarów pochłania znaczną ilość zasobów procesora oraz czasu. W celu minimalizacji zagrożeń z tej strony wykorzystałem moc obliczeniową klastra Prometheus, co umożliwiło badanie wydajności modeli oraz ewaluacja sieci bez obciążania własnego sprzętu.

1.4.2. Nieznajomość języka

Język Lua nie był mi wcześniej znany w praktyce, ale spędzenie czasu na nauce jego składni i zasad nie był czasem straconym, gdyż język jest bardzo ciekawy i w połączeniu ze środowiskiem Torch jest niebywale efektywny w dziedzinie uczenia maszynowego i obliczeń naukowych.

1.4.3. Nowa technologia

Jednym z większych wyzwań projektu było nauczenie się korzystania z ważniejszych bibliotek, które środowisko Torch udostępnia. Ta technologia również nie była mi wcześniej znana, ale widzę mnóstwo potencjalnych powodów, aby używać go w przyszłości.

1.4.4. Wymagający temat

Dość długi czas spędziłem próbując zrozumieć działanie RNN. Z pomocą przyszedł mi Alfredo Canzani (jeden z twórców Torch) ze swoją serią filmów Youtube na temat mechaniki sieci neuronowych, w szczególności tych rekurencyjnych. Studium kodu źródłowego biblioteki *char-rnn* Anthony'ego Karpathy'ego także dała mi sporo doświadczenia oraz pomysł na realizację końcowego produktu.

1.5. Studium wykonalności

Dzięki wykorzystaniu zasobów superkomputera Prometheus większość zagrożeń związanych z brakiem zasobów pamięciowych czy czasu procesora zostanie rozwiązane i w większym stopniu będę mógł skupić się na planowaniu i rozwijaniu projektu. Korzyściami wynikającymi z braku zespołu jest fakt, że czas nie jest tracony na konsultacje pomiędzy członkami oraz kod wykazuje większą spójność, jednakże projekt powstaje wolniej. Podsumowując, projekt ma duże szanse powodzenia, choć jego funkcjonalność może być ograniczona.

2. Zakres funkcjonalności

2.1. Wymagania funkcjonalne

- udostępnienie mechanizmu przetwarzania tekstów wejściowych w celu minimalizacji alfabetu języka

- zapewnienie eksportu danych do postaci binarnej, dającej możliwość wielokrotnego wczytania i wykorzystania ich bez ponownego preprocesingu
- możliwość zdefiniowania i stworzenia dowolnego modelu sieci rekurencyjnej określonego przez następujące parametry:
 - maksymalna ilość różnych autorów tekstów w modelu
 - wielkość warstwy ukrytej
 - poziom głębokości sieci
- parametryzowanie trenowania sieci poprzez:
 - rozmiar mini-batchów danych
 - współczynnik uczenia
 - tempo zmniejszania się współczynnika uczenia (redukcja następuje po przetworzeniu pojedynczej epoki)
 - maksymalną ilość epok
 - długość sekwencji znaków, na której uczony będzie model
- zapewnienie czytelnych logów, w trakcie działania programu, przedstawiających postęp uczenia oraz aktualny błąd uczenia.

2.2. Wymagania niefunkcjonalne

- realizacja projektu w ramach pracy inżynierskiej
- bazowanie na artykułach Douglasa Bagnalla na temat wielogłowych RNN i wykorzystanie napisanego przez niego modułu preprocesingu tekstu
- wykorzystanie języka Lua oraz środowiskaF Torch
- trening, testowanie i walidacja sieci na zbiorach PAN 2014 dla zadania weryfikacji autorstwa
- użycie systemu kontroli wersji Git

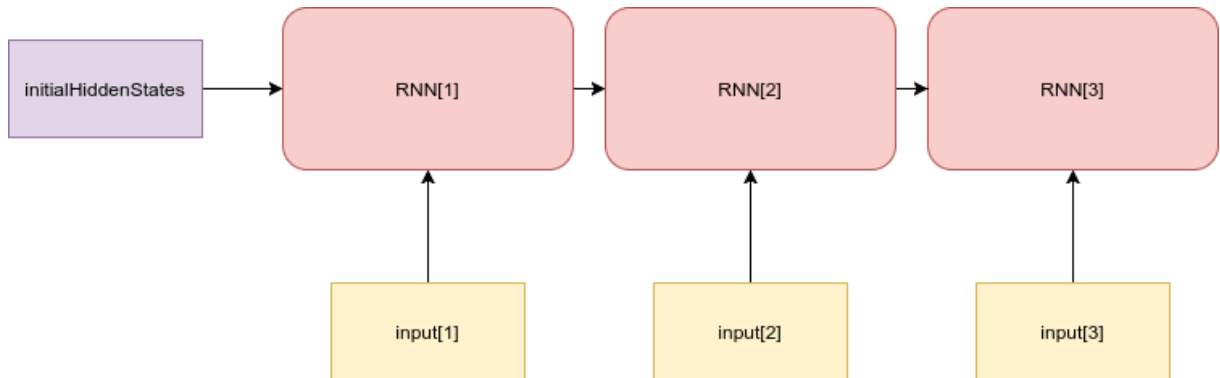
3. Wybrane aspekty realizacji

3.1. Wielogłowicowa głęboka rekurencyjna sieć neuronowa (ang. *Multi-headed deep RNN*)

Wielogłowicowa głęboka rekurencyjna sieć neuronowa różni się od tradycyjnej sieci typu feedforward tym, że zachowuje kontekst, tzn. poprzednie wejście sieci ma wpływ na jej aktualne wyjście. Pozwala to na uczenie jej na danych sekwencyjnych, takich jak tekst.

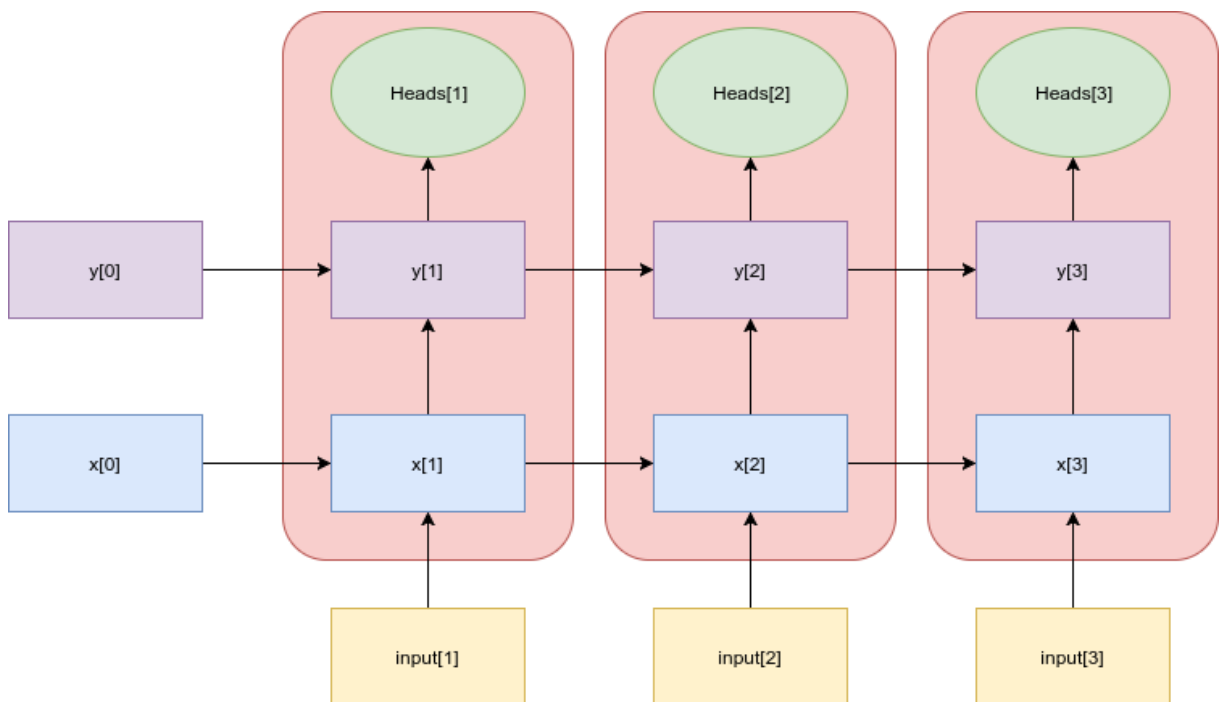
Aby nauczyć taką sieć, należy ją rozwinąć (ang. *unrolling*), tj. skopiować jej model i pospinać

ze sobą tak jak na rysunku 1, z zachowaniem zasady, że każda kopia współdzieli swoje parametry z każdą inną.



Rysunek 1: Płytki rekurencyjne sieci neuronowej

Wersja głęboka sieci rekurencyjnej polega na zwiększeniu ilości stanów ukrytych. Przykładowo na rysunku 2. przedstawiłem sieć o dwóch stanach ukrytych x i y . Jest to model wielogłowicowej głębokiej sieci rekurencyjnej o głębokości 2.



Rysunek 2: Wielogłowa głęboka rekurencyjna sieć neuronowa

Na sam koniec warto wspomnieć, co czyni sieć wielogłowicową. Otóż sieć może mieć wiele wyjść współdzielących podstawowy model RNN. W tym przypadku każde wyjście (każda głowica) reprezentuje prawdopodobieństwa wystąpienia kolejnego znaku dla danego autora w reakcji na aktualny znak na wejściu.

Za całość obsługi tych sieci odpowiada moduł *RNN* stworzonej biblioteki. Dostarcza on możliwości: tworzenia parametryzowanego modelu, rozwijania sieci, uczenia rozwiniętego modelu,

zapisu oraz odczytu już wytrenowanej sieci.

W odróżnieniu od sieci Bagnalla nie używałem funkcji aktywacji ReSQRT, lecz tanh. Algorytm uczenia w moim przypadku to zwykły SGD (ang. *Stochastic Gradient Descent*), a nie jego optymalizacja Adagrad.

3.2. Paczki danych

Aby przyspieszyć operację uczenia sieci, należy zastosować paczki danych (ang. *mini-batch*) danych. Jeżeli wprowadzamy na wejście sieci wektor, to otrzymujemy na wyjściu wektor, lecz równie dobrze można wprowadzić na wejście macierz złożoną z kilku wektorów i w efekcie otrzymać macierz na wyjściu. Takie podejście jest o wiele bardziej wydajne niż proste iterowanie po danych.

Za przygotowywanie mini-batchów danych odpowiedzialny jest moduł *BatchManager* stworzonej biblioteki.

3.3. Preprocessing

Alfabet tekstu wejściowego jest zbyt duży, co w sieciach rekurencyjnych może skutkować słabą generalizacją. W tym celu stosuję różne redukcje:

- każdą wielką literę zastępuję specjalnym prefiksem oraz odpowiadającą mu małą literą
- wszystkie nawiasy sprowadzam do jednej formy
- znaki, które w danym języku występują ze znikomą częstotliwością są usuwane
- oraz inne zależne od języka (angielski, grecki, holenderski, hiszpański), które rozwiązują specyficzne problemy związane z alfabetem, jak interpretacja liter arabskich w tekstach greckich.

Dodatkowo wielokrotne wystąpienia danego znaku pod rząd skracane jest do pięciu wystąpień.

3.4. Algorytm weryfikacji autorstwa

Rekurencyjna sieć neuronowa w teorii powinna lepiej przewidywać przebieg tekstu autora tekstów w zbiorze treningowym niż innych. Wielogłowa sieć posiada wiele głowic, każda z nich odpowiada za jednego autora. Dla tekstu wejściowego każda z nich produkuje współczynnik weryfikacji, który jest liczbą z przedziału $[0,1]$. Jeżeli wynik weryfikacji dla sprawdzanego autora jest większy niż średnia arytmetyczna wszystkich wyników, to tekst jest uznawany za napisany przez owego autora. W przeciwnym wypadku tekst sprawdzany autor nie jest uznawany za twórcę nieznanego tekstu.

4. Organizacja pracy

4.1. Metodyka

Praca nad projektem przebiegała w modelu iteracyjnym. Można wyróżnić najważniejsze części każdej iteracji:

- wybranie funkcjonalności
- implementacja
- testowanie

Po zebraniu wymagań, ale przed przejściem do planowania, mogę wyróżnić dodatkową fazę - zbierania informacji, gdyż musiałem jeszcze nauczyć się języka Lua, obsługi biblioteki Torch oraz teorii działania rekurencyjnych sieci neuronowych.

4.2. Postępy

W wyniku zastosowanego modelu oraz hierarchii funkcjonalności podjąłem się następujących iteracji:

1. definiowanie modelu - napisanie metody umożliwiającej utworzenie skonfigurowanego parametrami modelu głębokiej sieci RNN
2. wizualizacja oraz rozwinięcie modelu - dodanie opcji eksportu modelu w postaci grafu oraz zaimplementowanie metody rozwijającej model w czasie
3. podstawowy preprocessing - przygotowanie systemu przygotowującego tekst, poprzez redukcję rozmiaru alfabetu i zapis w postaci binarnej (na razie tylko dla języka angielskiego)
4. trening - zaimplementowanie metody uczenia rozwiniętej sieci
5. zapis oraz testy na Prometheusie - dodanie opcji eksportu i importu sieci oraz rozpoczęcie testów na klastrze
6. wielogłowa sieć - rozwinięcie sieci do postaci wielogłowej, poprawa implementacji, wprowadzenie możliwości uczenia na tekstach wielu autorów
7. testy na Prometheusie - testowanie wielogłowych sieci
8. wprowadzenie mini-batchów - implementacja batchera, która przyspieszyła naukę niemal trzykrotnie dla mini-batchów wielkości 40
9. pozostałe języki - dodanie wsparcia dla języków: hiszpańskiego, greckiego i holenderskiego
10. opcje i kolorowanie - parsowanie argumentów linii poleceń w celu uzyskania parametrów sieci oraz kolorowanie tekstu w konsoli

11. zaimplementowanie programu walidującego nauczone sieci
12. refaktoryzacja, modularyzacja oraz testowanie kodu

5. Wyniki projektu

5.1. Osiągnięcia

W wyniku przeprowadzonych prac udało się zaimplementować bibliotekę pozwalającą na tworzenie wielogłowych, głębokich sieci neuronowych - ich struktura jest kontrolowana parametrami, trenowanie sieci na specjalnie przygotowanych plikach uczących, zapisywanie wytrenowanych sieci oraz testowanie ich wydajności i skuteczności.

5.2. Problemy

5.2.1. Eksplodujący gradient

W trakcie treningu sieci wielokrotnie doświadczyłem zjawiska eksplodującego gradientu (ang. *vanishing gradient*), który wywołany był nieodpowiednim dobraniem współczynnika uczenia (ang. *learning rate*). Najlepszym sposobem na ominięcie tego problemu okazały się wielokrotne próby z różnymi parametrami, aby dobrać optymalny, domyślny współczynnik, który pozwoliłby na efektywne uczenie modeli.

5.2.2. Optymalizacja Adagrad

Wielokrotnie próbowałem wprowadzić optymalizację algorytmu SGD (ang. *stochastic gradient descent*), niestety wykorzystanie pakietu *optim* powodowało, że modele przestały się uczyć, co mogło być wynikiem specyficzności wielogłowych rekurencyjnych sieci neuronowych lub współdzielenia wag przez RNN.

5.3. Wyniki eksperymentów

Nawet sieci uczone krótko (nie więcej niż 5 epok) wykazywały skuteczność weryfikacji autorstwa relatywnie wysoką. Ewaluacji dokładności weryfikacji nieznanych tekstów w języku angielskim przedstawiona jest w tabeli 1.

Tabela 1: Wyniki eksperymentów dla tekstów w języku angielskim

Hidden size	Stacks	Heads	Mini-batch size	Epochs	Learning rate	BPTT steps	Verification score
199	3	10	40	5	0.002	20	0.9 (9/10)
199	3	20	40	5	0.010	20	0.8 (16/20)
99	2	10	80	5	0.005	50	0.8 (8/10)

5.4. Propozycje dalszego rozwoju

Sieci definiowane przez program używają zwykłego algorytmu mini-batch SGD (ang. *Stochastic Gradient Descent*), aby poprawić ich wydajność można by dodać pewne optymalizacje, np. za pomocą pakietu *optim*. Dzięki zastosowaniu mini-batchów nauka przebiega o wiele szybciej, choć z wykorzystaniem karty graficznej i pakietów *cuda* lub *cunn* możliwe jest osiągnięcie przyspieszenia wielokrotnie większego. Można też wprowadzić lepszą metodę weryfikacji autorstwa, charakteryzującą się wyższą skutecznością.

6. Dodatki

6.1. Spotkania z opiekunem

14.03.2017

Spotkanie w celu omówienia tematu pracy. Zostałem zapoznany ze szczegółami oraz zostały mi przedstawione dokumenty będące podstawą tej pracy, jak również zostałem poinformowany o potrzebie nauczania się języka Lua oraz obsługi biblioteki Torch.

28.03.2017

Spotkanie w celu doprecyzowania szczegółów i odpowiedzenia na pytania odnośnie charakterystyki tematu. Zostały mi przedstawione dodatkowe moduły Torcha pozwalające na rozwinięcie projektu mniejszym nakładem pracy.

23.05.2017

Spotkanie prezentujące postępy w pracach nad projektem. Zostałem poinformowany o możliwości wykorzystania klastra Prometheus do testów sieci oraz zostałem dodany do grantu obliczeniowego zapewniającego dostęp do zasobów superkomputera. Uzgodniliśmy również wtedy utworzenie repozytorium w systemie GitHub w celu nadzorowania dalszych postępów prac. Projekt jest dostępny pod adresem <https://github.com/Mantagar/Author-identification>.

6.2. Spotkania z prowadzącym pracownię projektową

W VI semestrze odbyły się cztery spotkania pracowni projektowej.

1. Na pierwszym przedstawione zostały nam zasady zaliczenia przedmiotu oraz zostaliśmy poinformowani o wymaganiach odnośnie kolejnych zajęć.
2. Na drugim spotkaniu prezentowaliśmy wizję naszych prac.
3. Na trzecim spotkaniu prezentowaliśmy analizę ryzyka oraz studium wykonalności naszych projektów oraz wymagania funkcjonalne i нефункционалне.
4. Na czwartym i ostatnim spotkaniu prezentowaliśmy postępy prac nad projektami oraz prototyp funkcjonalny systemu.

Materialy źródłowe

- [1] Lua tutorial. <https://www.lua.org/>.
- [2] Pl-grid user manual. <https://docs.cyfronet.pl/pages/viewpage.action?pageId=4260592>.
- [3] Torch reference manual. <https://github.com/torch/torch7>.
- [4] Torch tutorial. <http://torch.ch/docs/tutorials.html>.
- [5] Zadanie weryfikacji autorstwa. <http://pan.webis.de/clef15/pan15-web/author-identification.html>, <http://pan.webis.de/clef14/pan14-web/author-identification.html>.
- [6] D. Bagnall. Author identification using multi-headed recurrent neural networks. <https://arxiv.org/abs/1506.04891>.
- [7] D. Bagnall. Authorship clustering using multi-headed recurrent neural networks. <https://arxiv.org/abs/1608.04485>.
- [8] Y. Bengio. How to construct deep recurrent neural networks. <https://arxiv.org/abs/1312.6026>.
- [9] A. Canziani. Torch video tutorials. <https://www.youtube.com/playlist?list=PLLHTzKZzVU9ebuL6DCclzI54MrPNFGqbW>.
- [10] L. Deng and D. Yu. *Deep Learning*. Now Publishers, 2014.