

**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

<b>1</b>	<b>Cel prac i wizja produktu</b>	<b>4</b>
1.1	Charakterystyka problemu . . . . .	4
1.2	Motywacja projektu . . . . .	4
1.3	Wizja produktu . . . . .	4
1.4	Analiza ryzyka . . . . .	4
1.4.1	Czas testowania . . . . .	4
1.4.2	Nieznajomość języka . . . . .	5
1.4.3	Nowa technologia . . . . .	5
1.4.4	Wymagający temat . . . . .	5
1.5	Studium wykonalności . . . . .	5
<b>2</b>	<b>Zakres funkcjonalności</b>	<b>5</b>
2.1	Wymagania funkcjonalne . . . . .	5
2.2	Wymagania нефункционалне . . . . .	6
<b>3</b>	<b>Wybrane aspekty realizacji</b>	<b>6</b>
3.1	Multi-headed deep RNN . . . . .	6
3.2	Batch danych . . . . .	7
3.3	Preprocessing . . . . .	7
<b>4</b>	<b>Organizacja pracy</b>	<b>8</b>
4.1	Metodyka . . . . .	8
4.2	Postępy . . . . .	8
<b>5</b>	<b>Wyniki projektu</b>	<b>9</b>

# 1. Cel prac i wizja produktu

## 1.1. Charakterystyka problemu

Zadanie weryfikacji autorstwa polega na zdeterminowaniu, czy autor małego 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, choć w praktyce zastosowanie takiego podejścia może się nie sprawdzić.

## 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 frameworków (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 (character level language model RNN). Jednakże wciąż brakuje bibliotek, które pozwoliłyby na trenowanie sieci dla zadania weryfikacji autorstwa tekstu pisanego. Otóż przedmiotem tej pracy jest napisanie takiej właśnie biblioteki.

## 1.3. Wizja produktu

Najważniejszym założeniem projektu jest dostarczenie biblioteki, która pozwoli na konfigurowalne tworzenie głębokich sieci typu RNN, trening, preprocessing danych wejściowych, persistencja 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 wartościami: głębokość sieci, wymiar wektorów wejściowych, ilość neuronów w warstwach ukrytych, wielkość 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.

## 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 sieć nawet niedużych rozmiarów pochłania znaczną ilość zasobów procesora oraz wcześniej wspomnianego czasu, który może dochodzić nawet do kilkunastu godzin. W celu minimalizacji zagrożeń z tej strony wykorzystałem moc obliczeniową klastra Prometheus, co pozwoliło na testy 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 z frameworkiem 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 framework 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 Torcha) ze swoją serią filmów Youtube na temat mechaniki sieci neuronowych, w szczególności tych rekurencyjnych. Studium kodu źródłowego 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 użyteczność może być ograniczona.

## 2. Zakres funkcjonalności

### 2.1. Wymagania funkcjonalne

- udostępnienie mechanizmu przetwożenia tekstów wejściowych w celu minimalizacji alfabetu języka
- zapewnienie eksportu danych do postaci binarnej, dającej możliwość wielokrotnego wczytania i wykorzystania ich bez ponownego preprocessingu
- możliwość zdefiniowania i stworzenia dowolnego modelu określonego przez parametry:
  - maksymalna ilość różnych autorów tekstów w modelu
  - wielkość warstwy ukrytej
  - poziom głębokości sieci
- parametryzowanie trenowania sieci poprzez:

- wielkość batchów danych
  - współczynnik uczenia
  - tempo zmniejszania się współczynnika uczenia
  - maksymalną ilość epok
  - długość sekwencji znaków, na której uczony będzie model
- zapewnienie czytelnych logów, w trakcie działania programu, przedstawiających postęp uczenia oraz aktualny błąd

## 2.2. Wymagania нефunkcjonalne

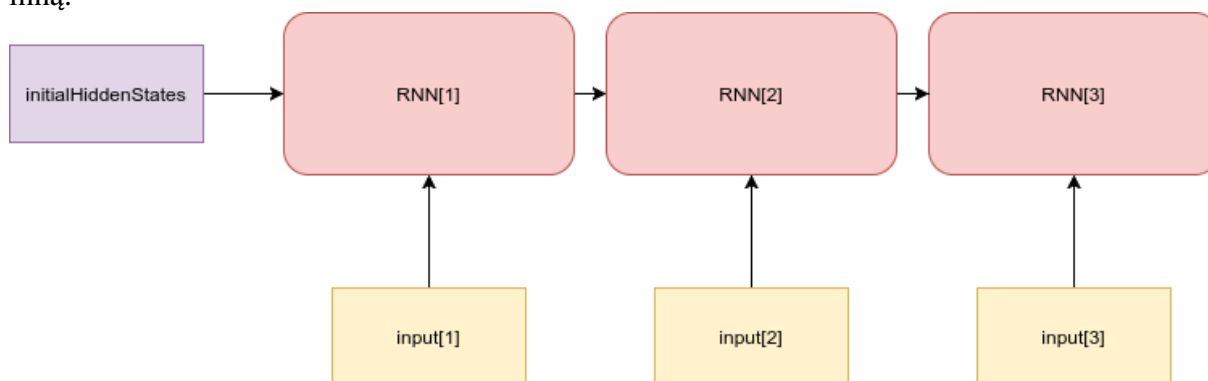
- realizacja projektu w ramach pracy inżynierskiej
- bazowanie na dokumentach Douglasa Bagnalla na temat wielogłowych RNN i wykorzystanie napisanego przez niego modułu preprocessingu tekstu
- wykorzystanie języka Lua oraz frameworka Torch
- trenowanie i testowanie sieci na zbiorach PAN 2014 dla zadania weryfikacji autorstwa
- użycie systemu kontroli wersji Git

## 3. Wybrane aspekty realizacji

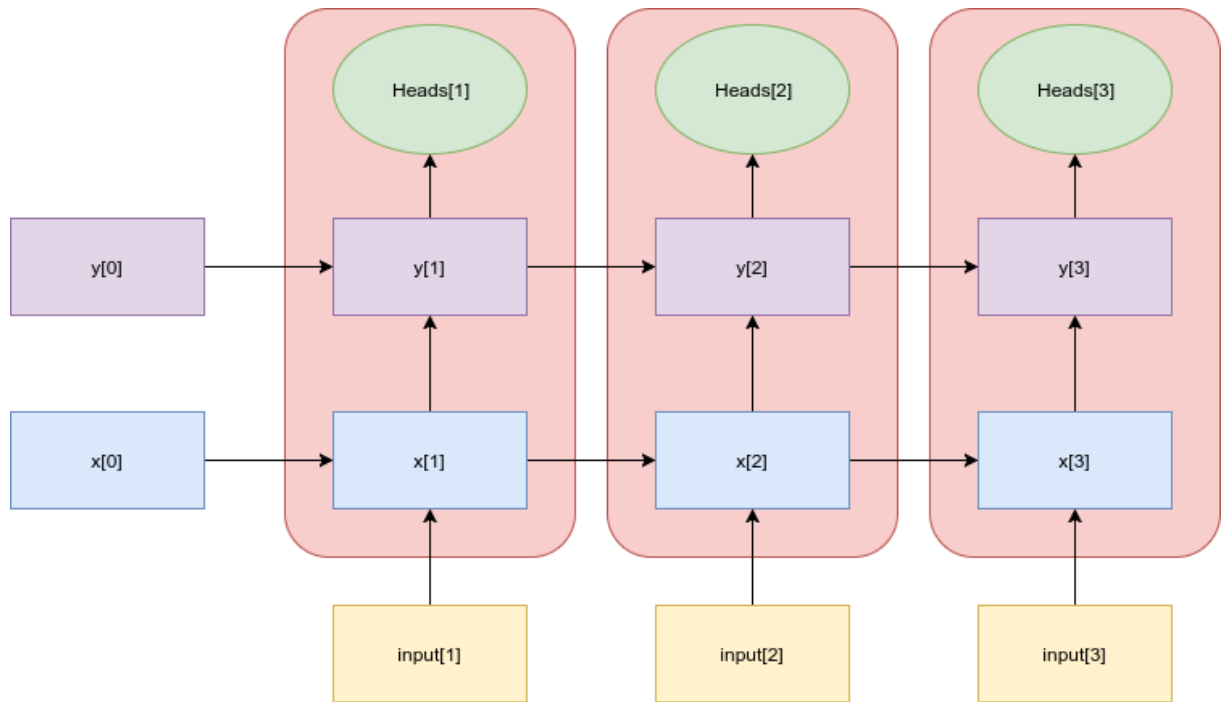
### 3.1. Multi-headed deep RNN

Rekurencyjna sieć neuronowa różni się od sieci MLP 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, tj. skopiować jej model i pospinać ze sobą tak jak na rysunku poniżej, z zachowaniem zasady, że każda kopia współdzieli swoje parametry z każdą inną.



Wersja głęboka sieci rekurencyjnej polega na zwiększeniu ilości stanów ukrytych. Przykładowo na rysunku poniżej przedstawiłem sieć o dwóch stanach ukrytych  $x$  i  $y$ .



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

Za całość obsługi tych sieci odpowiada moduł RNN, który jest sercem 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 normalny SGD, a nie jego optymalizacja Adagrad.

### 3.2. Batch danych

Aby przyspieszyć operację uczenia sieci, należy zastosować batche 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 batchów danych odpowiedzialny jest moduł BatchManager.

### 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
- żadkie znaki usuwam z alfabetu
- oraz inne zależne od języka (angielski, grecki, holenderski, hiszpański)

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

## 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 Torch oraz teorii działania RNN.

### 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 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 minibatchów - implementacja batchera, która przyspieszyła naukę niemal trzykrotnie minibatchów 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



## 5. Wyniki projektu

*Wskazanie wyników projektu (co konkretnie udało się uzyskać: oprogramowanie, dokumentacja, raporty z testów/wdrożenia, itd.), prezentacja wyników i ocena ich użyteczności (jak zostało to zweryfikowane — np. wnioski klienta/użytkownika, zrealizowane testy wydajnościowe, itd.), istniejące ograniczenia i propozycje dalszych prac.*

## Materialy źródłowe

- [1] Lua tutorial. <https://www.lua.org/>.
- [2] Podręcznik uŹytkownika pl-grid. <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] D. Bagnall. Author identification using multi-headed recurrent neural networks. <https://arxiv.org/abs/1506.04891>.
- [6] D. Bagnall. Authorship clustering using multi-headed recurrent neural networks. <https://arxiv.org/abs/1608.04485>.
- [7] A. Canziani. Torch video tutorials. <https://www.youtube.com/playlist?list=PLLHTzKZzVU9ebuL6DCclzI54MrPNFGqbW>.
- [8] L. Deng and D. Yu. *Deep Learning*. Now Publishers, 2014.