Uniwersytet im. Adama Mickiewicza w Poznaniu

Wydział Matematyki i Informatyki

Praca magisterska

# AUTOMATYCZNE TŁUMACZENIE POWIEŚCI WIZUALNYCH

Machine Translation of Visual Novels

SŁAWOMIR GOLIJASZ

Numer albumu: 396324

Kierunek: Informatyka

Promotor:

prof. UAM dr hab. Krzysztof Jassem

Poznań, 2018

Poznań, dnia .....

(data)

# OŚWIADCZENIE

Ja, niżej podpisany **Sławomir Golijasz** student Wydziału Matematyki i Informatyki Uniwersytetu im. Adama Mickiewicza w Poznaniu oświadczam, że przedkładaną pracę dyplomową pt: **Ąutomatyczne tłumaczenie powieści wizualnych"** napisałem samodzielnie. Oznacza to, że przy pisaniu pracy, poza niezbędnymi konsultacjami, nie korzystałem z pomocy innych osób, a w szczególności nie zlecałem opracowania rozprawy lub jej części innym osobom, ani nie odpisywałem tej rozprawy lub jej części od innych osób.

Oświadczam również, że egzemplarz pracy dyplomowej w wersji drukowanej jest całkowicie zgodny z egzemplarzem pracy dyplomowej w wersji elektronicznej.

Jednocześnie przyjmuję do wiadomości, że przypisanie sobie, w pracy dyplomowej, autorstwa istotnego fragmentu lub innych elementów cudzego utworu lub ustalenia naukowego stanowi podstawę stwierdzenia nieważności postępowania w sprawie nadania tytułu zawodowego.

[ ]*-wyrażam zgodę na udostępnianie mojej pracy w czytelni Archiwum UAM

[ ]*-wyrażam zgodę na udostępnianie mojej pracy w zakresie koniecznym do ochrony mojego prawa do autorstwa lub praw osób trzecich

*Należy wpisać TAK w przypadku wyrażenia zgody na udostępnianie pracy w czytelni Archiwum UAM, NIE w przypadku braku zgody. Niewypełnienie pola oznacza brak zgody na udostępnianie pracy.

.....

(czytelny podpis studenta)

**Streszczenie**

Celem niniejszej pracy magisterskiej jest porównanie statystycznego tłumaczenia maszynowego z neuronowym tłumaczeniem maszynowym w powieściach wizualnych z języka angielskiego na język polski i skonstruowanie prototypu systemu tłumaczenia maszynowego, który tłumaczy treść w skrypcie powieści wizualnych z języka angielskiego na język polski.

W pracy zawarto: wytłumaczenie pojęć powieść wizualna i tłumaczenie maszynowe, zagadnienia związane z tłumaczeniem statystycznym, tłumaczeniem neuronowym i oceną automatyczną, informacje związane z wykorzystanymi w pracy narzędziami, zasadę działania i budowę systemu stworzonego w ramach projektu oraz wyniki badań.

**Abstract**

The purpose of this thesis is to compare statistical machine translation with neural machine translation in visual novels from English to Polish and construct a prototype machine translation system, which translates the content in the script of visual novels from English to Polish.

The thesis includes: explanation of the term visual novel and machine translation, issues related to statistical translation, neural translation and automatic evaluation, information related to the tools used in the thesis, the principle of operation and the construction of the system created as part of the thesis project and research results.

# Spis treści

# Wstęp

Już w XVII wieku pojawił się pomysł stworzenia systemu tłumaczenia automatycznego, ale dopiero rozwój komputerów w XX wieku pozwolił przyczynić pierwsze kroki w tym kierunku. W 1954 roku na Uniwersytecie Georgetown został przedstawiony eksperymentalny system tłumaczenia automatycznego, jednak rozwój tej dziedziny nauki został mocno spowolniony po raporcie ALPAC (z ang. Automatic Language Processing Advisory Committee — Komitet Doradczy ds. Automatycznego Przetwarzania Języka), ponieważ dziesięcioletnie badania nad tłumaczeniem automatycznym nie spełniły oczekiwań. Dopiero w latach 80 prace nad tłumaczeniem automatycznym zostały wznowione, gdyż moc obliczeniowa komputerów stała się wystarczająca do udźwignięcia problemów związanych z tłumaczeniem automatycznym.

W dzisiejszych czasach istnieje wiele systemów tłumaczenia automatycznego, które służą do ogólnego użytku lub do tłumaczenia konkretnych rodzajów zbiorów tekstowych. Jednak jeszcze nikt do tej pory nie podjął się stworzenia systemu tłumaczenia automatycznego powieści wizualnych.

Jedyna praca najbliższa temu tematowi ([VSHT10]) jest o tłumaczeniu napisów filmowych w kontekście tłumaczenia statystycznego. Z powodu, że ostatnio tłumaczenie neuronowe wypiera tłumaczenie statystyczne, podjąłem się sprawdzenia, jak te dwa różne systemy tłumaczenia automatycznego radzą sobie z tłumaczeniem powieści wizualnych.

Celem niniejszej pracy jest porównanie statystycznego tłumaczenia maszynowego z neuronowym tłumaczeniem maszynowym w powieściach wizualnych z języka angielskiego na język polski i skonstruowanie prototypu systemu tłumaczenia maszynowego, który tłumaczy treść w skrypcie powieści wizualnych z języka angielskiego na język polski.

Pierwszy rozdział w niniejszej pracy tłumaczy najważniejsze pojęcia związane z pracą magisterską. Rozdział drugi, trzeci i czwarty omawia teoretyczne zagadnienia związane odpowiednio z tłumaczeniem statystycznym, tłumaczeniem neuronowym i oceną automatyczną. Rozdział piąty przedstawia narzędzia użyte w projekcie magisterskim. Szósty rozdział omawia sposób przeprowadzenia projektu i wyniki badań. Na końcu znajduje się Podsumowanie.

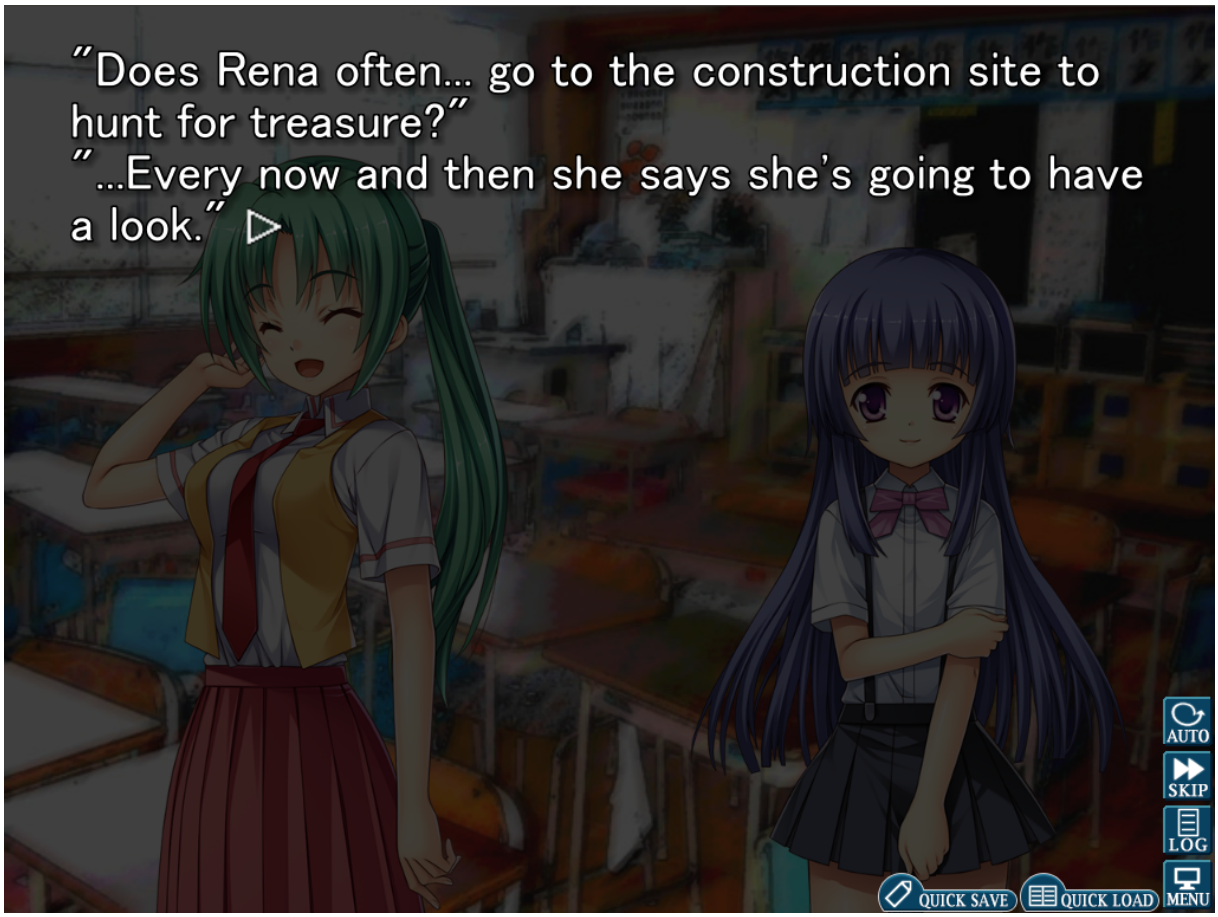
# Rozdział 1

# Wytłumaczenie pojęć

## 1.1 Powieść wizualna

Powieść wizualna (ang. *Visual Novel*) jest to gra komputerowa, charakteryzująca się multimedialnym przedstawieniem najczęściej obszernej, nieliniowej fabuły z pierwszoosobową narracją wraz ze statyczną grafiką lub animacją w stylu *anime* (jest to w krajach zachodnich określenie na japońskie filmy animowane lub styl animacji japońskiej), podkładem muzycznym i niekiedy głosowym.

Powieść wizualna (Rys. 1.1) działa na podobnej zasadzie co gry paragrafowe (gry książkowa polegająca na czytaniu opisów i dokonywaniu wyborów, z których każdy odsyła nas do innego paragrafu w książce). Gracz w powieści wizualnej co jakiś czas będzie musiał dokonać wyboru, a fabuła historii będzie się rozgałęziać w zależności od jego decyzji. Często zdarza się, że przy dokonaniu wielu błędnych wyborów bohater powieści wizualnej ginie niefortunną śmiercią, zmuszając gracza do użycia zapisu gry i ponownej próby przejścia historii. W powieści wizualnej nie wszystkie wybory są pokazywane graczowi. Gra na podstawie jego wcześniejszych decyzji pokazuje możliwe rozgałęzienia historii. Na przykład w *Fate/Stay Night* bohater Shiro w jednej ze ścieżek powieści jest poważnie ranny i bliski śmierci. Jeśli w trakcie gry był wystarczająco przyjazny dla Ilyi (dziewczyny, z którą spotyka się w różnych momentach w historii), przyjdzie mu na ratunek. Jeśli nie, będzie musiał sam się o siebie zatroszczyć. Inne funkcje, takie jak możliwość zapisu i szybkiego przejścia przez rozdziały, które zostały już przeczytane, ułatwiają graczom pełne poznanie historii i wypróbowanie różnych ścieżek. Będąc produktem cyfrowym, powieści wizualne nie napotykają na ograniczenia długości występujące w książkach fizycznych [LK11].




Rys. 1.1: Przykład powieści wizualnej na podstawie powieści wizualnej *Higurashi no naku koro ni*. Źródło: gra komputerowa *Higurashi no naku koro ni*

## 1.2 Tłumaczenie maszynowe

Tłumaczenie maszynowe lub tłumaczenie automatyczne (ang. *machine translation*) polega na zautomatyzowaniu tłumaczenia tekstu lub mowy z jednego języka na drugi. Mówiąc inaczej, jest to tłumaczenie tekstu lub mowy z języka źródłowego ("source") na język docelowy ("target"). Przyjęło się stosować poniższe oznaczenia na język źródłowy i docelowy:

- **f**-język źródłowy
- **e**-język docelowy

W tłumaczeniu maszynowym stosuje się korpus równoległy. Korpusem równoległym nazywamy zbiór odpowiadających sobie tekstów dla co najmniej dwóch różnych języków. W niniejszej pracy korpusem równoległym określana jest para odpowiadających sobie tekstów w języku źródłowym i docelowym.

Sam pomysł zautomatyzowania tłumaczenia pojawił się już na początku XVII wieku, ale dopiero pojawienie się komputerów umożliwiło rozwój tego zagadnienia z dziedziny przetwarzania języka naturalnego (ang. *natural language processing*). Do lat 80 dominowały systemy regułowe (ang. *rule-based*

*machine translation*), których działanie opiera się na regułach stworzonych manualnie. W latach 90 zaczął się okres panowania systemów statystycznych, nazywanych także systemami tłumaczenia statystycznego (ang. *statistical machine translation*), które zaczęły wykorzystywać korpusy równoległe i metody statystyczne do tłumaczenia tekstu. W ostatnich latach systemy tłumaczenia neuronowego (ang. *neural machine translation*) stały się popularne z powodu lepszych wyników w stosunku do systemów tłumaczenia statystycznego. Wykorzystują one sieć neuronową do tłumaczenia tekstu lub mowy z języka źródłowego na docelowy [Koz14].

# Rozdział 2

# Tłumaczenie statystyczne

Tłumaczenie statystyczne skupia się na wyniku [JD08], czyli istotne jest, aby uzyskane tłumaczenie było możliwe najbardziej prawdopodobnym odpowiednikiem zdania źródłowego. Założenie te zostało sformalizowane w równaniu (2.1).

$$e_{najlepszy} = argmax_e P(E = e|F = f) \tag{2.1}$$

gdzie E i F są zmiennymi losowymi, które przebiegają odpowiednio po wszystkich zdaniach języka źródłowego i docelowego. Dla wybranego zdania f zdanie $e_{najlepszy}$ maksymalizuje powyższą funkcję i przez to jest najbardziej prawdopodobnym tłumaczeniem zdania f [JD08].

Oprócz tego, że tłumaczenie powinno być wierne, powinno także być płynne. Oznacza to, że powinno oddawać sens zdania źródłowego jak najdokładniej, będąc przy tym poprawną albo co najmniej zrozumiałą wypowiedzią w języku docelowym. Takie ujęcie sugeruje pewną modularność problemu tłumaczenia. Łatwiej jest zamodelować dwie oddzielne funkcje prawdopodobieństwa dla każdego kryterium, czyli funkcję, która mierzy wierność zdania docelowego względem zdania źródłowego oraz funkcję mierzącą poprawność i płynność zdania docelowego. Metody statystycznego tłumaczenia działają dokładnie w taki sposób [JD08]. Wystarczy tylko przekształcić wzór (2.1) za pomocą reguły Bayesa – wzór (2.2).

$$P(E = e|F = f)\frac{P(F = f|E = e)P(E = e)}{P(F = f)} \tag{2.2}$$

$$e_{najlepszy} = argmax_e P(F = f|E = e)P(E = e) \tag{2.3}$$

Równanie (2.3) nazywa się *podstawowym równaniem tłumaczenia statystycznego* i jest jednym ze szczególnych przypadków zaszumionego kanału. Mianownik P(F=f) ze wzoru (2.2) został pominięty, ponieważ przy ustalonym f nie wpływa na wybór najbardziej prawdopodobnego e. Mamy tutaj dwa komponenty. Pierwszy z nich to model tłumaczenia P(F=f|E=e), a drugi to model języka P(E=e) [JD08].

Najprościej wytłumaczyć działanie systemu tłumaczenia statystycznego na zasadzie głuchego telefonu. Załóżmy, że przy zabawie uczestniczą dzieci znające dwa języki, na przykład język polski i angielski, i ostatnie dziecko ma za zadanie odgadnąć, jakie było początkowe zdanie w języku angielskim na podstawie otrzymanego zdania w języku polskim. Przebieg zabawy wygląda następująco: pierwsze dziecko wymyśla i szepcze zdanie w języku angielskim, ale w czasie przekazywania zdania po angielsku jedno z dzieci uczestniczących w zabawie zmienia je na zdanie w języku polskim. Ostatnie dziecko stara się odgadnąć na podstawie otrzymanego zdania polskiego, jakie było początkowe zdanie w języku angielskim. I tak dziecko wymyślające zdanie angielskie reprezentuje model języka, dziecko zmieniające zdanie w języku angielskim na zdanie w języku polskim reprezentuje model tłumaczenia, natomiast ostatnie dziecko odpowiada maszynie dekodującej. Język angielski i polski to odpowiednio język źródłowy i docelowy.

## 2.1 Model tłumaczenia

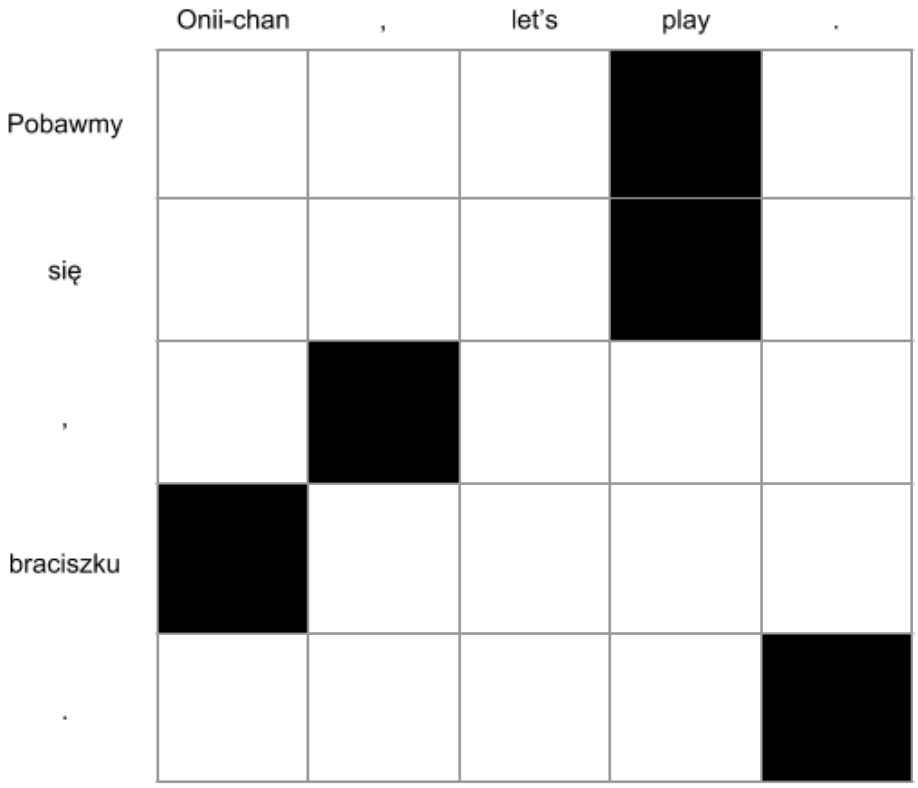
Rozróżniamy dwa modele tłumaczenia w systemach tłumaczenia statystycznego. Są to: model oparty na wyrazach (ang. *word-based*) i model oparty na frazach (ang. *phrase-based*). W obu z nich wykorzystuje się funkcję dopasowania (ang. *alignment function*).

Model oparty na wyrazach najpierw dzieli zdanie na wyrazy, które później są dopasowywane za pomocą funkcji dopasowania.

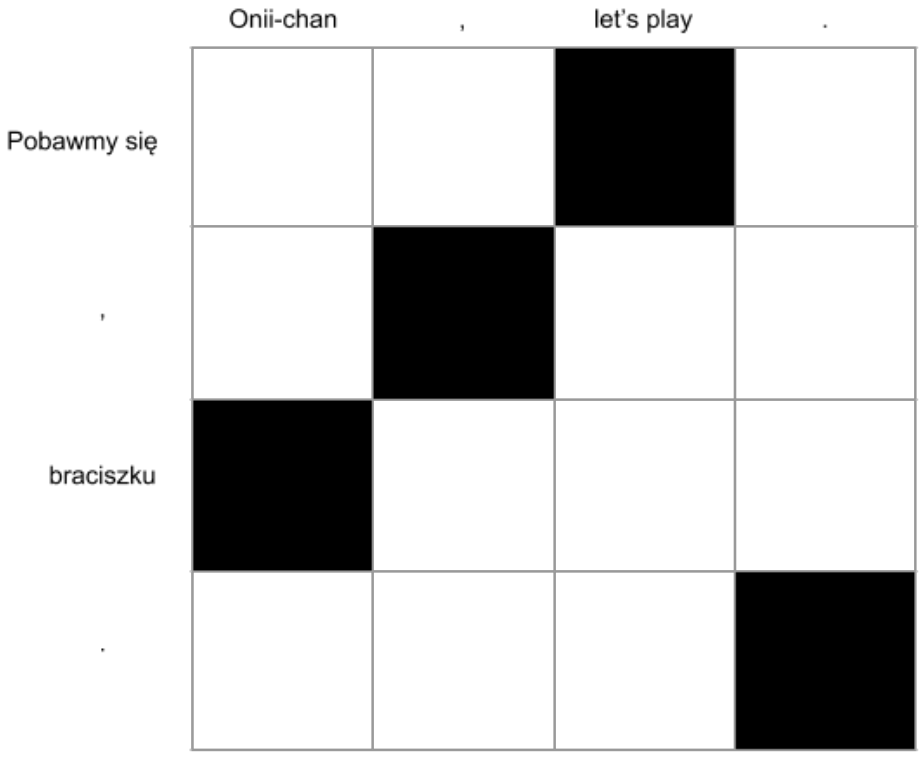
Rys. 2.1 przedstawiono idealne dopasowanie wyrazów w modelu opartym na wyrazach dla zdania w języku angielskim w powieści wizualnej do zdania w języku polskim. Wyraz *let's* nie został dopasowany, gdyż nie ma odpowiednika wyrazowego w zdaniu w języku polskim.

Model oparty na frazach działa na podobnej zasadzie, tylko że zdanie jest dzielone na frazy zamiast na wyrazy. Frazą może być wyraz, zbiór wyrazów lub całe zdanie.

Rys. 2.2 przedstawiono idealne dopasowanie wyrazów w modelu opartym na frazach dla zdania w języku angielskim w powieści wizualnej do zdania w języku polskim. Wyraz *let's* teraz nie został pominięty, a połączył się z wyrazem *play*, które razem tworzą frazę odpowiadającą frazie *pobawmy się* w języku polskim. Dzięki modelowi opartemu na frazach można uniknąć wiele błędów w tłumaczeniu w sytuacjach, w których wyrazy tworzące frazę w języku źródłowym i docelowym oznaczają osobno całkowicie inne pojęcie niż jako fraza (Przykład 2.1).




Rys. 2.1: Przykład dopasowania w modelu opartego na wyrazach.




Rys. 2.2: Przykład dopasowania w modelu opartego na frazach.

| Eng | Pl |
|---|---|
| It | To |
| is | jest |
| raining | pada |
| cats | koty |
| and | i |
| dogs | psy |

Przykład 2.1: Przykład dosłownego tłumaczenia angielskiego odpowiednika zdania "Leje jak z cebra".

## 2.2 Model języka

Ważnym elementem każdego systemu tłumaczenia statystycznego jest model języka, który mierzy, jak bardzo prawdopodobne jest, że sekwencja wyrazów byłaby wypowiedziana przez rodzimego użytkownika języka [Koe10a]. Na przykład model języka otrzymał zdania w języku polskim "idę do szkoły" i "do idę szkoły". Zmierzy on prawdopodobieństwo obu zdań i większe prawdopodobieństwo otrzyma zdanie "idę do szkoły", bo bardziej prawdopodobne jest, że rodzimy użytkownik języka polskiego powie pierwsze zdanie:

$$P_{LM}(\textit{idę do szkoły}) > P_{LM}(\textit{do idę szkoły})$$

Najlepszą metodą modelowania modelu języka jest podejście n-gramowe. N-gramowy model języka bazuje na zamodelowaniu, jak prawdopodobne jest, że słowa następują po sobie. Przeważnie stosuje się trigramowy model języka, a do jego oszacowania stosuje się wzór 2.4, gdzie:

- $count(w_1, w_2, w_3)$ – *liczba wystąpień trigramu* $w_1, w_2, w_3$ *w korpusie języka docelowego*
- $\sum_w \quad count(w_1, w_2, w)$ – *liczba wystąpień wszystkich trigramów z początkowym ciągiem wyrazów* $w_1, w_2$ *w korpusie języka docelowego*
- $P(w_3|w_1w_2)$ – *prawdopodobieństwo wystąpienia wyrazu* $w_3$ *po ciągu wyrazów* $w_1, w_2$

$$p(w_3|w_1, w_2) = \frac{count(w_1, w_2, w_3)}{\sum_w \quad count(w_1, w_2, w)} \tag{2.4}$$

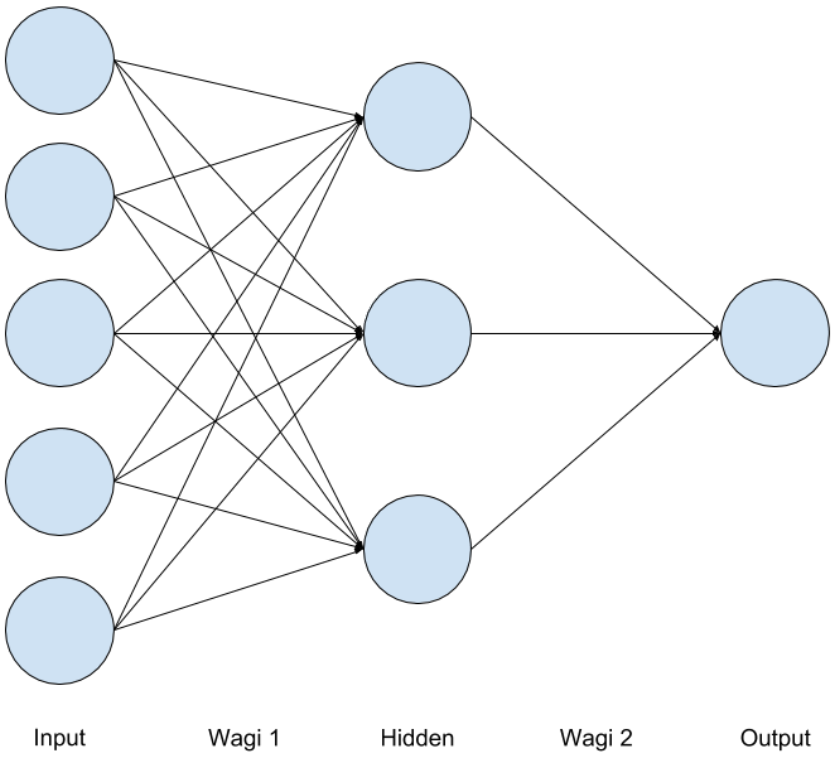
Wadą n-gramowego modelu języka jest, że przy n-gramie niewystępującym w korpusie, poda wartość zero. W celu rozwiązania tego problemu stosuje się wygładzanie (ang. *smoothing*).

Wygładzanie dzielimy na metody ogólne np. wygładzanie jedynką (ang. *add-one smoothing*) [Koe10a] i wygładzanie sekwencyjne np. wygładzanie przez cofanie (ang. Back-off) [Koe10a] lub interpolacja (ang. Interpolation) [Koe10a].

# Rozdział 3

# Tłumaczenie neuronowe

Tłumaczenie neuronowe wykorzystuje sieć neuronową (Rys. 3.1). Sieć neuronowa jest to technika uczenia maszynowego, która pobiera szereg danych wejściowych i przewiduje dane wyjściowe.

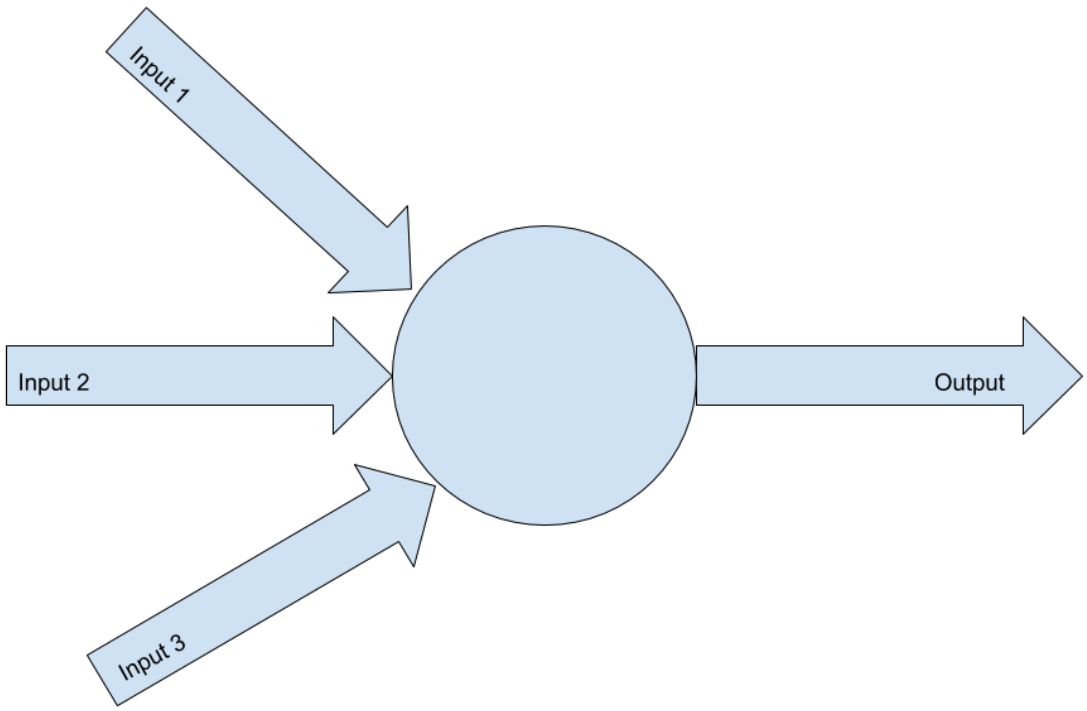



Rys. 3.1: Zobrazowanie sieci neuronowej na przykładzie sieci jednokierunkowej.

Sieci neuronowe używają wielu warstw. Zamiast obliczać wartość wyjściową bezpośrednio z danych wejściowych, wprowadzona jest warstwa ukryta (ang. *hidden layer*). Jest nazywana ukrytą, ponie-

waż możemy obserwować dane wejściowe i wyjściowe sieci neuronowej, ale nie możemy obserwować działania mechanizmu sieci neuronowej pomiędzy warstwą wejściową (ang. *input layer*) i warstwą wyjściową (ang. *output layer*).

Warstwa składa się z neuronów (Rys. 3.2), które nie łączą się między sobą, a łączą się z neuronami w sąsiadujących warstwach. Każde "połączenie" między neuronami posiada wagę.




Rys. 3.2: Uproszczony sztuczny neuron.

Sieci neuronowe wymagają optymalizacji wartości wag, aby prawidłowo przewidywały wynik. Używa się do tego zbioru treningowego. Wielokrotnie wprowadza się dane wejściowe ze zbioru treningowego do sieci neuronowej. Następnie porównuje się wyliczone dane wyjściowe z sieci neuronowej z poprawnymi danymi wyjściowymi ze zestawu treningowego i aktualizuje się wagi. Zazwyczaj przeprowadza się kilka razy dane treningowe przez sieć neuronową. Każde takie przekazanie danych nazywa się powtórzeniem (ang. *epoch*).

Najczęściej stosowaną metodą uczenia sieci neuronowych jest propagacja wsteczna (ang. *back-propagation*). Dla każdego przykładu ze zbioru treningowego najpierw aktualizuje wagi dla warstwy wyjściowej. Następnie propaguje informację o błędzie do wcześniejszych warstw i aktualizuje wagi w kierunku od warstwy wyjściowej do warstwy wejściowej.

## 3.1 Neuronowy model języka

Słowa w neuronowym modelu języka są reprezentowane przez wektory. Wykorzystuje się do tego typ wektora o nazwie *1 z n* (ang. *one hot vector*), który ma *n* współrzędnych, w których jedna ma wartość 1, a pozostałe mają wartość 0. Do każdego słowa przypisuje się konkretny wektor. Na przykład:

- *Onii-chan* = (0,1,0,0,0,0,0,0)
- *play* = (0,0,0,0,0,0,0,1)
- *let's* = (0,0,0,0,0,0,1,0)

Oprócz reprezentacji słów przez wektory używa się w neuronowym modelu języka osadzania słowa (ang. *word embedding*). Działanie osadzania słowa polega na generalizacji słów po przez przesłanki wynikające ze zdań. Na przykład, jeśli dane treningowe dla modelu języka często zawierają n-gramy:

- *onii-chan play with me*
- *onee-chan play with me*
- *i love my onii-chan*
- *i love my onee-chan*
- *my onii-chan is the best*
- *my onee-chan is the best*

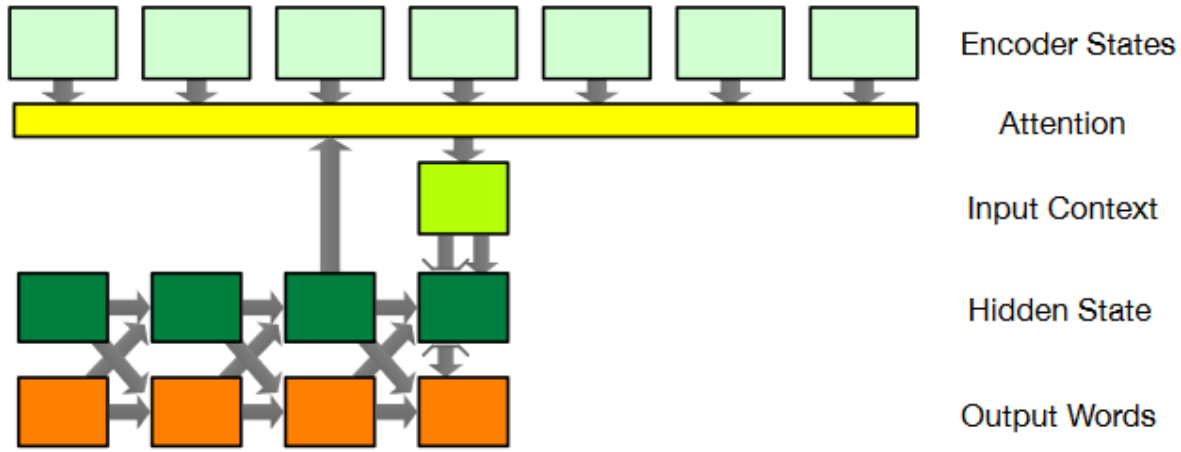
wtedy model języka skorzysta z wiedzy, że onii-chan i onee-chan występują w podobnych kontekstach, a zatem są w pewnym stopniu zamienne. Skutkiem generalizacji słów poprzez osadzanie słowa jest posiadanie rzetelniejszych przewidywań słów [Koe17].

## 3.2 Model tłumaczenia neuronowego z podejściem kodowania-odkodowywania

Model tłumaczenia z podejściem kodowania-odkodowywania (ang. *translation models encoder-decoder approach*) polega na tym, że korzystamy z dwóch sieci neuronowych. Pierwsza sieć koduje dane z korpusu z języka źródłowego na wektory, które przekazywane są drugiej sieci neuronowej, która odkodowywuje je, jako zdania w języku docelowym.

W celu powiązania reprezentacji słów pomiędzy siecią kodującą a dekodującą stosuje się mechanizm atencji (Rys. 3.3). Na Rys. 3.3 mechanizm atencji jest informowany przez wszystkie reprezentacje słów

wejściowych i poprzedni stan ukryty (ang.*hidden state* [Koe17]) sieci neuronowej dekodującej, tworząc określenie kontekstu tłumaczenia.




Rys. 3.3: Zobrazowanie działania mechanizmu uwagi. Źródło: [Koe17]

Do poradzenia sobie z dużym słownikiem i nieznanymi słowami stosuje się jednostki podwyrazowe (ang. *subword units*). Jest to metoda, która polega na podziale słów w korpusie, na przykład *website* na *web* i *site*.

Ta metoda jest trenowana na korpusie równoległym. Najpierw za każdym słowem wstawia się specjalne oznaczenie, które oznacza koniec słowa. Potem łączy się najczęściej występującą parę znaków i traktuje się ją jako jeden znak (spacja nie jest traktowana jako znak). Ten krok powtarza się ustaloną ilość razy.

# Rozdział 4

# Ocena automatyczna

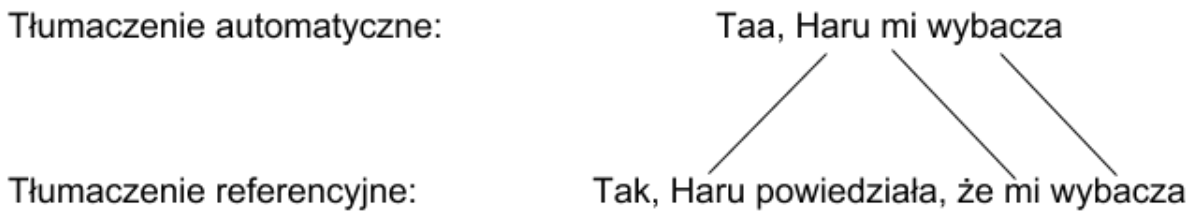
Ocena automatyczna (ang. *Automatic Evaluation*) jest jedną z metod do oceny systemu tłumaczenia automatycznego. Jest szybsza i tańsza od oceny ludzkiej wykonanej przez zawodowych tłumaczy. System tłumaczenia automatycznego jest porównywany z jednym lub wieloma ludzkimi tłumaczeniami – nazywanymi także tłumaczeniami referencyjnymi (ang. *reference translation*) – tego samego zdania. Im bardziej zdanie przetłumaczone przez system tłumaczenia automatycznego jest podobne do zdania z tłumaczenia referencyjnego, tym bardziej jest prawdopodobne, że jest ono poprawne.

Jedną z metryk do automatycznej oceny systemu tłumaczenia automatycznego jest precyzja (ang. precision). Porównuje liczbę poprawnie przetłumaczonych wyrazów na podstawie tłumaczenia referencyjnego do długości tłumaczenia wykonanego przez system tłumaczenia automatycznego (Wzór 4.1).

$$\text{precyzja} = \frac{\text{liczba wyrazów występująca w obu tłumaczeniach}}{\text{liczba wyrazów występująca w zdaniu tłumaczenia automatycznego}} \tag{4.1}$$

Precyzja dla przykładu z Rys. 4.1 wynosi 3/4, ponieważ trzy wyrazy występują zarówno w tłumaczeniu automatycznym, jak i referencyjnym, a zdanie z tłumaczenia automatycznego ma cztery wyrazy.

Wadą metryki precyzji jest, że faworyzuje krótkie zdania. Wystarczy stworzyć zdanie z jednego najczęściej pojawiającego się wyrazu w języku docelowym, żeby uzyskać precyzję równą 1 dla większości zdań.




Rys. 4.1: Przykład dopasowania wyrazów między tłumaczeniem automatycznym a tłumaczeniem referencyjnym.

Inną z metryk do automatycznej oceny systemu tłumaczenia automatycznego jest BLEU (A Bilingual Evaluation Understudy). Jest to najpopularniejsza metryka oceny automatycznej. Do ocenienia tłumaczenia wykonanego przez system automatyczny stosuje dopasowania n-gramowe do wyliczenia precyzji n-gramowej. Precyzja n-gramowa jest obliczana podobnie jak metryka precyzji, z tym że zamiast wyrazu występuje n-gram.

Przykład 4.1 jest na podstawie Rys. 4.1. Precyzja unigramowa wynosi 3/4, ponieważ trzy unigramy (*Haru*, *mi*, *wybacza*) występują zarówno w tłumaczeniu automatycznym, jak i referencyjnym, a zdanie z tłumaczenia automatycznego ma w sumie cztery unigramy. Precyzja bigramowa wynosi 1/3, bo tylko jeden bigram występuje zarówno w tłumaczeniu automatycznym, jak i referencyjnym, a całkowita ilość bigramów w tłumaczeniu automatycznym wynosi trzy (*Taa Haru*, *Haru mi*, *mi wybacza*). Precyzja trigramowa wynosi 0/2, ponieważ nie ma wspólnego trigramu dla tłumaczenia automatycznego i referencyjnego, a w tłumaczeniu automatycznym mamy tylko dwa trigramy (*Taa Haru mi*, *Haru mi wybacza*).

$$\text{BLEU-n} = \text{kara-za-zwięzłość}\ exp \sum_{i=1}^{n} \lambda_i \log \text{precyzja}_i \tag{4.2}$$

$$\text{kara-za-zwięzłość} = min\Big(1, \frac{\text{długość zdania tłumaczenia automatycznego}}{\text{długość zdania tłumaczenia referencyjnego}}\Big) \tag{4.3}$$

Metryka BLEU rozwiązuje problem metryki precyzji — brak kary za pominięcie wyrazów — przez karę za zwięzłość (ang. *brevity penalty*). Kara zmniejsza wynik, jeśli zdanie tłumaczenia automatycznego jest za krótkie (Formuła 4.3). Zazwyczaj stosuje się n-gramy długości 4. Taka metryka BLEU nazywana jest BLEU-4. Wagi $\lambda_i$ (Formuła 4.2) dla różnych precyzji zwykle ustawia się na wartość jeden, co pozwala uprościć formułę 4.2 do formuły 4.4 [Koe10b].

$$\text{BLEU-n} = min\Big(1, \frac{\text{długość zdania tłumaczenia automatycznego}}{\text{długość zdania tłumaczenia referencyjnego}}\Big) \prod_{i=1}^{n} \text{precyzja}_i \tag{4.4}$$

Przykład 4.1 przedstawia wyniki BLEU po zastosowaniu formuły 4.4 dla przykładu z Rys. 4.1. Kara za zwięzłość wynosi 4/6, ponieważ długość zdania tłumaczenia automatycznego wynosi 4 (*Taa*, *Haru*, *mi*, *wybacza*), a długość zdania tłumaczenia referencyjnego wynosi 6 (*Tak*, *Haru*, *powiedziała*, *że*, *mi*, *wybacza*). Stosując formułę 4.4 dla metryk BLEU-1, BLEU-2 i BLEU-3 uzyskuje się ocenę BLEU odpowiednio 0.5, 0.17 i 0.

| Metryka | wynik dla przykładu Rys. 4.1 |
|---|---|
| Precyzja (1-gram) | 3/4 |
| Precyzja (2-gram) | 1/3 |
| Precyzja (3-gram) | 0/2 |
| Kara za zwięzłość | 4/6 |
| BLEU-1 | 0.5 |
| BLEU-2 | 0.17 |
| BLEU-3 | 0 |

Przykład 4.1: Przykład wyników BLEU dla przykładu z Rys. 4.1

# Rozdział 5

# Narzędzia użyte w projekcie

## 5.1 System do tłumaczenia neuronowego Marian-nmt

System Marian jest efektywnym zestawem narzędzi do neuronowego tłumaczenia automatycznego napisanym w czystym języku programowania C++. Wspiera on trenowanie i tłumaczenie wielu popularnych modeli NMT i został opracowany na Uniwersytecie im. Adama Mickiewicza w Poznaniu (UAM) oraz na Uniwersytecie w Edynburgu. System Marian jest wciąż rozwijany, dzięki czemu jest aktualnym i efektywnym narzędziem do neuronowego tłumaczenia automatycznego. Kolejną jego zaletą jest to, że jest udostępniony na licencji MIT, która daje prawo do nieograniczonego używania oprogramowania.

Preferowanym systemem operacyjnym jest Ubuntu w wersji 16.04 LTS lub Ubuntu w wersji 14.04 LTS. Do działania oprogramowania dla wersji 16.04 wymagane są: narzędzia programistyczne CMake 3.5.1 i GCC/G++ 5.4, biblioteka Boost 1.58, architektura CUDA 8.0. Dla wersji 14.04 są to: narzędzia programistyczne CMake 3.5.1 i GCC/G++ 4.9, biblioteka Boost 1.54, architektura CUDA 8.0. Instalacja i obsługa odbywa się za pomocą poleceń w konsoli systemu operacyjnego. Pełna lista instrukcji i sposób instalacji znajduje się w dokumentacji na stronie marian-nmt.github.io.

Trenowanie jest obsługiwane tylko przez procesory GPU i dodatkowo jest potrzebne do tego architektura Nvidia CUDA, która jest obsługiwane przez pewne karty graficzne od Nvidii. Samo tłumaczenie tekstu nie stanowi problemu dla obecnych komputerów, ponieważ można wykorzystać do tego dowolne procesory CPU. Aktualna wersja zestawu narzędzi jest dostępna w systemie *Github*. Czas instalacji samego zestawu narzędzi trwa około godziny, a czas trenowania jest zależny od wielkości korpusu treningowego i mocy obliczeniowej stacji roboczej.

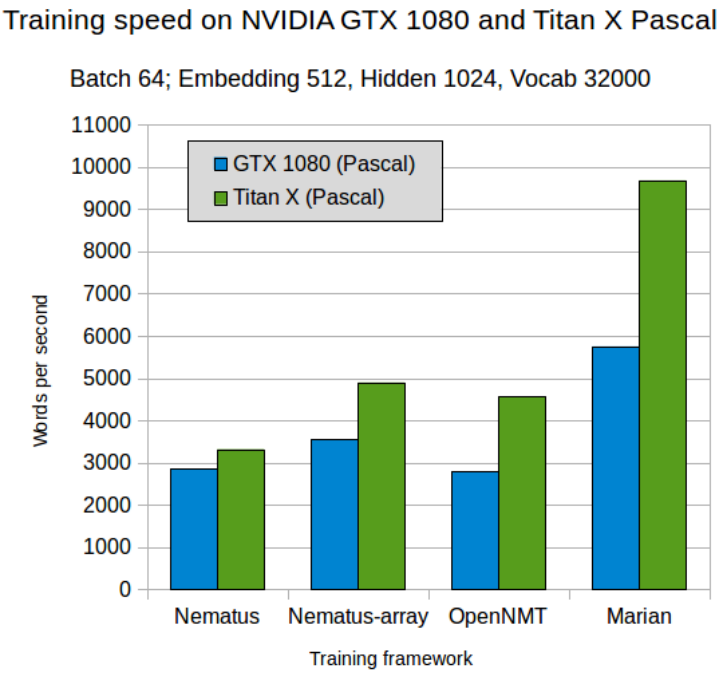
### 5.1.1 Architektura CUDA

Architektura CUDA jest równoległą architekturą obliczeniową opracowaną przez firmę Nvidia, która zapewnia wzrost wydajności obliczeniowej przez wykorzystanie mocy układów GPU. Jest ona dostępna dla kart graficznych GeForce, ION Quadro i Tesla.

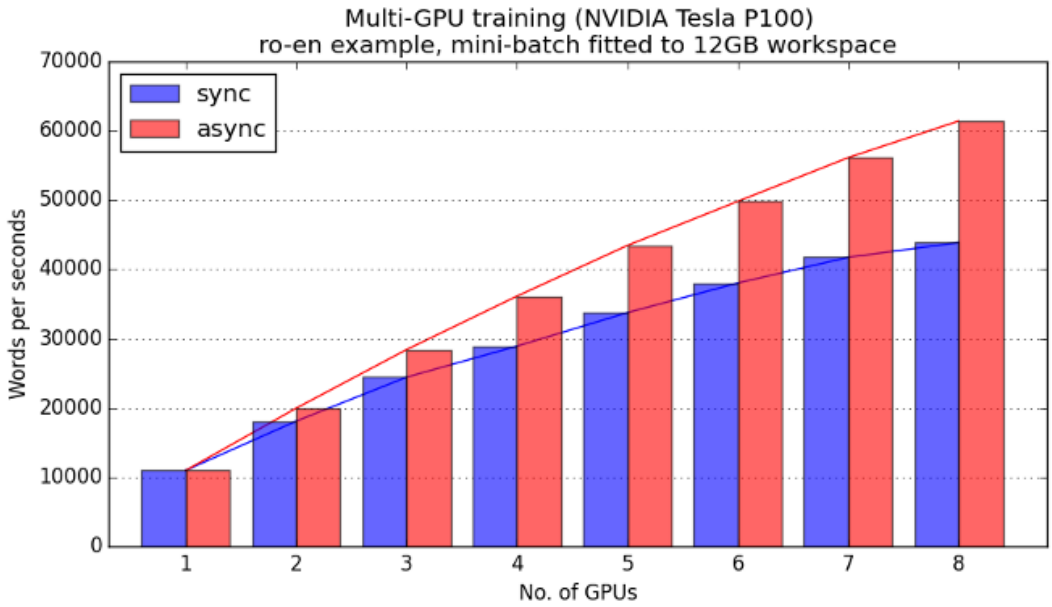
### 5.1.2 Trenowanie

System Marian-nmt jest oparty na modelu encoder-decoder. Do wytrenowania modelu do tłumaczenia z jednego języka do drugiego potrzebny jest korpus równoległy. Proces ten jest czasochłonny, jednak system Marian jest bardzo wydajnym narzędziem do trenowania systemu neuronowego. Na Rys. 5.1 ukazano różnice prędkości trenowania dla różnych systemów neuronowego tłumaczenia automatycznego.

Prędkość trenowania można polepszyć przez wykorzystanie większej liczby kart GPU. Służy do tego opcja –devices, która pozwala wybrać, które z kart graficznych wezmą udział w trenowaniu. Dodatkowo można ustawić algorytm stochastyczny spadku wzdłuż gradientu (ang. *Stochastic Gradient Descent*, w skrócie SGD) na pracę synchroniczną za pomocą opcji –sync-sgd, która zwiększa prędkość trenowania. Domyślnie algorytm SGD jest ustawiona na pracę asynchroniczną. Na Rys.5.2 ukazano wzrost prędkości tłumaczenia wraz ze wzrostem liczby procesorów GPU przy opcjach *sync* i *ansync*.




Rys. 5.1: Porównanie prędkości trenowań. Źródło: `https://marian-nmt.github.io/features/`

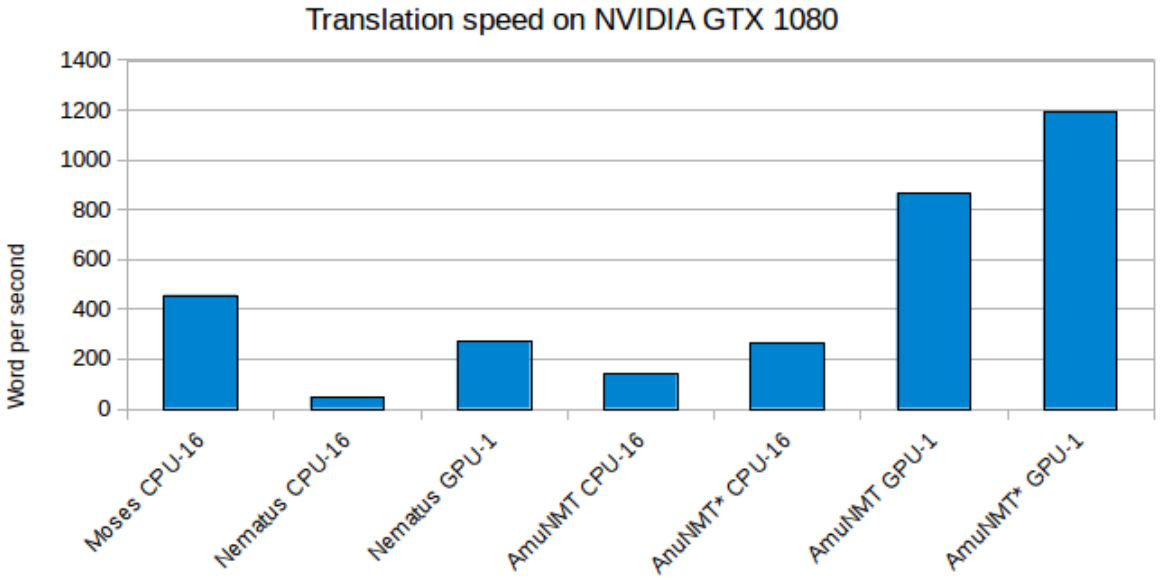



Rys. 5.2: Trenowanie na wielu GPU. Źródło: `https://marian-nmt.github.io/features/`

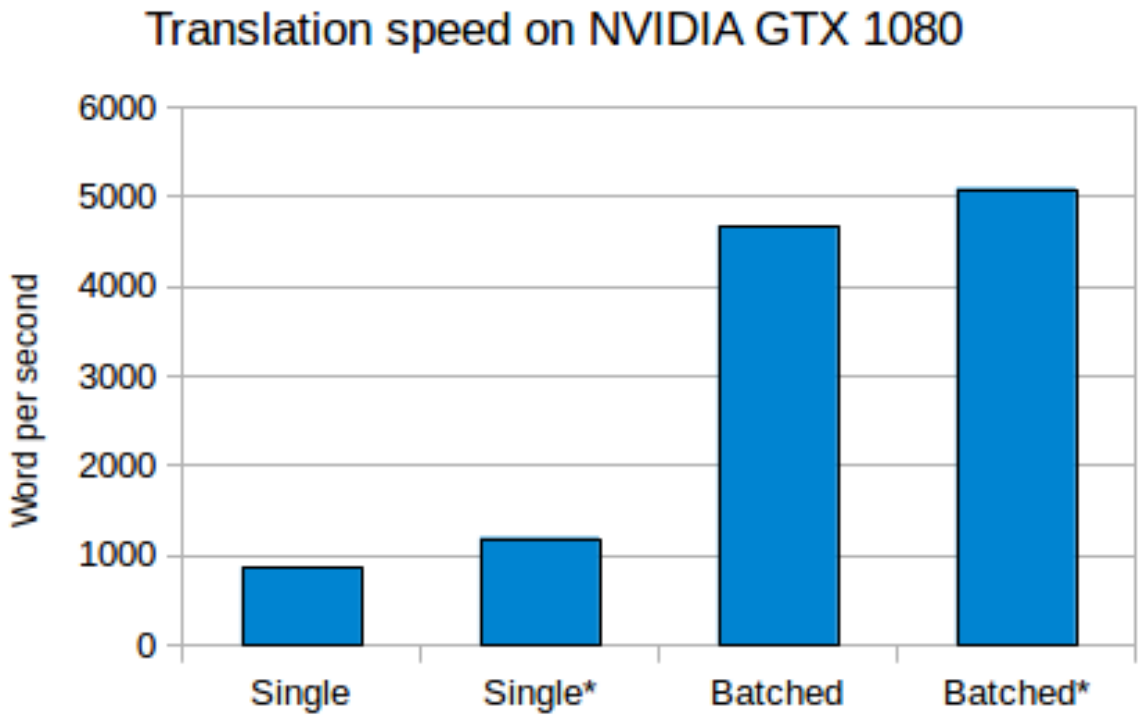
### 5.1.3 Tłumaczenie

Do tłumaczenia można użyć narzędzia Marian decoder (narzędzie w systemie Marian-nmt służące do tłumaczenia tekstu z języka źródłowego do docelowego) lub narzędzie Amun (inne narzędzie w systemie Marian-nmt służące do tłumaczenia tekstu z języka źródłowego do docelowego). Narzędzie Marian decoder wspiera jedynie tłumaczenie przy pomocy procesorów GPU. W przeciwieństwie do niego system Amun jest w stanie tłumaczyć przy pomocy procesorów CPU lub GPU. Na Rys. 5.3 przedstawiono prędkości tłumaczeń dla poszczególnych modeli przy użyciu procesorów CPU lub GPU.

Narzędzie Amun oferuje również tłumaczenie w trybie wsadowym takim, w którym wiele zdań jest tłumaczonych jednocześnie na pojedynczym procesorze GPU. Dzięki temu możemy osiągnąć o wiele większą prędkość tłumaczenia słów. Na Rys. (5.4) ukazano prędkości tłumaczenia dla trybu wsadowego.




Rys. 5.3: Porównanie prędkości tłumaczeń dla poszczególnych modeli. Źródło: `https://marian-nmt.github.io/features/`




Rys. 5.4: Porównanie prędkości tłumaczeń AmuNMT w trybie wsadowym i bez. Źródło: `https://marian-nmt.github.io/features/`

## 5.2 System do tłumaczenia statystycznego Moses

System Moses jest narzędziem do statystycznego tłumaczenia automatycznego, który umożliwia stworzenie translatora tłumaczącego z jednego języka na drugi. Wymagane do tego jest posiadanie dużego korpusu równoległego. Wciąż jest rozwijany i jest na licencji LGP, która pozwala na jego darmowe użytkowanie. System Moses może być wykorzystywany w systemach Linux, Windows i Mac Os. Aktualną wersję systemu Mosesa można pobrać w systemie *Github*. Cały proces instalacji i użytkowania przebiega za pomocą poleceń w konsoli w powyższych systemów operacyjnych. Wszystkie polecenia i tutoriale można znaleźć na stronie systemu Moses.

Obecnie każdy komputer spełnia wymagania do korzystania z systemu Moses. Instalacja narzędzia Moses zajmuje kilka godzin, a sam czas trenowania danych jest zależny od wielkości korpusów równoległych i mocy obliczeniowej komputera.

### 5.2.1 Trenowanie

Proces trenowania w Mosesie polega na stworzeniu modelu statystycznego tłumaczenia. Składa się on z 9 kroków, które są napisane w języku Perl w programie train-model.perl. Są nimi:

1. Przygotowanie danych
2. Uruchomienie programu GIZA++
3. Dopasowanie słów
4. Otrzymanie tabeli tłumaczeń leksykalnych
5. Ekstrakcja fraz
6. Punktowanie fraz
7. Zbudowanie modelu porządku
8. Zbudowanie generatora modelu
9. Stworzenie pliku konfiguracyjnego

Przygotowanie danych konwertuje korpus równoległy do formatu właściwego dla narzędzia GIZA++. W tym narzędziu są zaimplementowane ogólnodostępne modele IBM, które wykonują operacje trenowania tłumaczenia automatycznego metodą statystyczną. Następny jest krok dopasowania słów, przy czym domyślną metodą jest Grow-Diag-Final. Kolejnym krokiem jest otrzymanie tabeli tłumaczeń leksykalnych, które bierze pod uwagę wyrównanie w poprzednim kroku i szacuje prawdopodobieństwa

dopasowania słowa w języku X do słowa w języku Y w obu kierunkach, tworząc tabelę tłumaczeń leksykalnych. Po tym następuje krok ekstrakcji fraz, gdzie wydobywa się wszystkie frazy i umieszcza się w jednym dużym pliku. Następnym krokiem jest punktowanie fraz, które jest podobne do działania kroku otrzymania tabeli tłumaczeń leksykalnych. Następnie jest tworzony model porządku, który oblicza koszt liniowy do porządku odległości (na przykład pomijanie dwóch słów jest bardziej kosztowne niż pominięcie jednego). W kroku zbudowania generatora modelu jest tworzony ostateczny model automatycznego tłumaczenia statystycznego. Ostatni krok – stworzenie pliku konfiguracyjnego – generuje plik konfiguracyjny dla dekodera ze wszystkimi prawidłowymi ścieżkami do wygenerowanego modelu i domyślnymi wartościami ustawień parametrów.

# Rozdział 6

# Projekt Magisterski

## 6.1 Cel projektu

Celem projektu jest porównanie tłumaczenia statystycznego z tłumaczeniem neuronowym przy tłumaczeniu powieści wizualnych z języka angielskiego na język polski i skonstruowanie prototypu systemu tłumaczenia automatycznego skryptów powieści wizualnych z języka angielskiego na język polski.

## 6.2 Gromadzenie i przygotowanie danych do uczenia maszynowego

Dane do trenowania i oceniania systemów automatycznego tłumaczenia powieści wizualnych z języka angielskiego na polski zostały zebrane za pomocą strony internetowej *The Visual Novel Database*.
Dla ułatwienia pozyskania danych z powieści wizualnych skupiono się na powieściach wizualnych opartych na trzech najpopularniejszych silnikach gry do powieści wizualnej (*KiriKiri*, *NScripter*, *Ren'Py*).

Po pobraniu z internetu tytułów powieści wizualnych posiadających tekst w języku angielskim i polskim ekstrahuje się z nich treść. W tym celu użyto autorskich skryptów napisanych w języku Pythona (Rys.6.1) wykorzystujących wyrażenia regularne do wydobycia treści z powieści wizualnej.

```
1 #!/usr/bin/python3
2 import re
3 import glob
4 import pathlib
5 import argparse
6
```

```
# regex
regex = re. compile(r'(;;)?(\[.+])?(.+)((\[np])|(\[wvl]))')

# args
parser = argparse.ArgumentParser()
parser.add_argument("path",  type= str,
                    help="path of working")
parser.add_argument("visual_novel_directory",  type= str,
                    help="path of text
                    from visual novel scipt")
parser.add_argument("text_directory",  type= str,
                    help="directory of text
                    from visual_novel")
args = parser.parse_args()

# funkction
for _file in glob.glob(args.path + "/"
                       + args.visual_novel_directory
                       + "/*.ks"):
    name = _file.split(sep='/')[-1].split(sep='.')[0]
    pathlib.Path(args.path + "/" + args.text_directory)
        .mkdir(parents=True, exist_ok=True)
    out =  open(args.path + "/" + args.text_directory + '/'
                + name + '.txt', 'w', encoding='utf-8')

    with  open(_file, encoding='utf-16') as f:
        contents = f.read()

    sentence = re.findall(regex, contents)
    for each in sentence:
        if not each[0]:
            out.write(each[2] + '\n')

    f.close()
```

```
41      out.close()
```

Rys. 6.1: Przykład skryptu dla pozyskania tesktu z plików zawierających treść powieści wizualnej dla silnika KiriKiri.

Następnie dokonuje się dopasowania tekstu, ponieważ treści w różnych językach w powieściach wizualnych nie zawsze są dopasowane. W tym celu zastosowano program *Hunalign*, który jest programem do dopasowywania tekstów dwujęzycznych. Na Rys.6.2 przedstawiono działanie programu *Hunalign*. W celu usunięcia znaków połączenia wierszy przez program *Hunalign* zastosowano autorski skrypt do ich usunięcia.

```
532 It's said that a studious person can never go back to hard work once they learn the pleasure of leisure.
533 Similarly, rediscovering one's conscience halfway down the path of darkness wildly shakes one's illusion of control.
534 That night, I had a hard time sleeping.
```

```
532 Tak jak pracowita osoba nigdy nie powróci do ciężkiej pracy, gdy raz pozna przyjemność lenistwa, tak samo osobie, która raz wkroczy na złą drogę, ciężko jest z niej zawrócić nawet jeśli na nowo odkryje swoje sumienie.
533 Tej nocy nie mogłem zasnąć.
```

```
532 It's said that a studious person can never go back to hard work once they learn the pleasure of leisure. ~~~ Similarly, rediscovering one's conscience halfway down the path of darkness wildly shakes one's illusion of control.  Tak jak pracowita osoba nigdy nie powróci do ciężkiej pracy, gdy raz pozna przyjemność lenistwa, tak samo osobie, która raz wkroczy na złą drogę, ciężko jest z niej zawrócić nawet jeśli na nowo odkryje swoje sumienie. -0.00491803
533 That night, I had a hard time sleeping. Tej nocy nie mogłem zasnąć. 0.1875
```

Rys. 6.2: Przykład dopasowania tekstu przez zastosowanie programu *Hunalign*.

Kolejnym krokiem przygotowania danych z powieści wizualnych do trenowania i oceniania wyników systemu automatycznego tłumaczenia powieści wizualnych jest podział danych na zbiory testowe i zbiór trenujący. Następnie zawartości zbiorów są rozdzielane na tekst w języku angielskim i w języku polskim. Na koniec odbywa się usunięcie znaczników tekstowych w zbiorze trenującym i w zbiorach testowych, które mogą wystąpić w tekście (Rys.6.3). Do ich usunięcia zastosowano autorskie skrypty w języku Python dla poszczególnych silników gry (Rys.6.4).

```
1  Jest dopiero późny październik, a już można poczuć na twarzy podmuch północnego wiatru.[l] W promieniach zachodzącego słońca, droga stopniowo traci swój kolor. Za rogiem wielkiego budynku, w bocznej uliczce, spadające liście tańczą na wietrze.[l] Mrużąc oczy, aby zapobiec dostaniu się do nich pyłu, ludzie rozbiegają się do swoich domów.
```

Rys. 6.3: Przykładowe znaczniki z powieści wizualnej opartej na silniku KiriKiri we fragmencie wyekstrahowanego tekstu.

```
1 #!/usr/bin/python3
2 # -*- coding: utf-8 -*-
3 import sys
```

```
import re


for line in sys.stdin:
    print(re.sub(r"(\{i\})|(\{/i\})|(\{size=.+\})
                 |(\{/size\})|(\{p=.+\})|(\{nw\})
                 |(\{b\})|(\{/b\})|(\{w=.+\})", "",
                  line)[:-1])
```

Rys. 6.4: Przykład skryptu usuwającego znaczniki tekstowe z gry silnika Ren'Py.

Przedostatnim krokiem jest oczyszczenie zbioru trenującego zawierającego dane z powieści wizualnych z pustych wierszy i nadmiarowej spacji. W tym celu wykorzystano program z systemu Moses o nazwie *clean-corpus-n* (Rys.6.5).

```
${MOSES_SCRIPTS_DIR}/training/clean-corpus-n.perl\
   ${MAIN_PATH}/translator_data_vn/corpus/text ${SRC} ${TRG}\
   ${MAIN_PATH}/translator_data_vn/corpus/clear_text 1 200
```

Rys. 6.5: Polecenie uruchomienia programu *clean-corpus-n* do wyczyszczenia korpusu.

Ostatnim krokiem jest tokenizacja zbioru trenującego. Użyto do tego programu zawartego w systemie Moses o nazwie *tokenizer* (Rys.6.6).

```
#!/bin/bash
LANG=$1

: ${NUM_THREADS:=8}
: ${MOSES_SCRIPTS_DIR:=/home/deinonzch/Pulpit/mosesdecoder/\
   scripts}

${MOSES_SCRIPTS_DIR}/tokenizer/tokenizer.perl\
   -threads ${NUM_THREADS} -l ${LANG}
```

Rys. 6.6: Skrypt bashowy uruchamiający program *tokenizer* do tokenizacji danych.

Oprócz danych z powieści wizualnych użyto także danych ze strony internetowej *OpenSubtitles2018*. Do ich przygotowania skorzystano ze skryptu napisanego w języku Python do wyciągnięcia tekstu i następnie skryptu języka Python do ich dopasowania. (Rys.6.7 i Rys.6.8). Następnie dopasowany tekst z *OpenSubtitles2018* został podzielony na tekst w języku angielskim i tekst w języku polskim. Na koniec dane z *OpenSubtitles2018* zostały stokenizowane przez program *tokenizer* ze systemu Moses (Rys.6.6). Dane z *OpenSubtitles2018* zostały wykorzystane do zbioru trenującego w niektórych modelach systemu automatycznego tłumaczenia powieści wizualnych.

```
1  import re
2  import sys
3
4  regex = re. compile(r'<seg>(.+)</seg>')
5
6  for line in sys.stdin:
7      words = re.findall(regex, line)
8      if words:
9          print(words[0])
```

Rys. 6.7: Skrypt do pozyskania tekstu z pliku ze strony internetowej *OpenSubtitles2018*.

```
1  #!/usr/bin/python3
2  # -*- coding: utf-8 -*-
3  import sys
4  import argparse
5
6  # args
7  parser = argparse.ArgumentParser()
8  parser.add_argument("target", type= str,
9                      help="target file")
10 args = parser.parse_args()
11
12 english_sentence = []
13 polish_sentence = []
14
15 i = 1
```

```
16 for line in sys.stdin:
17     if i == 2:
18         polish_sentence.append(line)
19         i = 1
20     else:
21         english_sentence.append(line)
22         i = 2
23
24 train_tsv =  open(args.target, 'w', encoding='utf-8')
25
26 for num_line in  range( len(polish_sentence)):
27     train_tsv.write(english_sentence[num_line][:-1]
28                     + '\t' + polish_sentence[num_line])
29
30 train_tsv.close()
```

Rys. 6.8: Skrypt do dopasowania tekstu z pliku ze strony internetowej *OpenSubtitles2018* po jego wydobyciu.

## 6.3 Trenowanie

Do stworzenia modeli do tłumaczenia powieści wizualnych z języka angielskiego na polski użyto systemów Moses i Marian, które odpowiednio trenują model do tłumaczenia statystycznego i model do tłumaczenia neuronowego.

Do wytrenowania modelu tłumaczenia statystycznego powieści wizualnych w systemie Moses są potrzebne oczyszczone korpusy w języku polskim i angielskim oraz model języka. Do stworzenia modelu języka w systemie Moses potrzebny jest korpus w języku docelowym (korpus w języku polskim). Model języka trenuje się w systemie Moses przez program *lmplz* (Rys.6.9). W celu szybszego wczytywania modelu języka należy utworzyć jego wersję binarną. Do tego służy w systemie Moses program *build_binary* (Rys.6.10). Model tłumaczenia statystycznego w systemie Moses trenuje się za pomocą programu *train-model* (Rys.6.11). W celu szybszego wczytywania modelu tłumaczenia należy go zbinaryzować. Do utworzenia wersji binarnej modelu tłumaczenia służy w systemie Moses program *processPhraseTableMin* (Rys.6.12). Na końcu trzeba ustawić ścieżkę do wersji binarnej modelu tłumaczenia dla systemu Moses (Rys.6.13).

```
${MOSES_BIN_DIR}/lmplz  -o 5 -S 80% < $< > $@
```

Rys. 6.9: Polecenie uruchomienia programu *lmplz* do wytrenowania modelu języka.

```
${MOSES_BIN_DIR}/build_binary -i $< $@
```

Rys. 6.10: Polecenie uruchomienia programu *build_binary* do stworzenia modelu języka w wersji binarnej.

```
${MOSES_SCRIPTS_DIR}/training/train-model.perl \
   --root-dir translator_data_vn/arena \
   -external-bin-dir ${MGIZA_BIN_DIR} \
   -mgiza -mgiza-cpus ${NUM_THREADS} -cores ${NUM_THREADS} \
   -corpus translator_data_vn/corpus/preprocessed -f ${SRC} \
   -e ${TRG} -alignment grow-diag-final-and \
   -lm 0:3:${MAIN_PATH}/translator_data_vn/corpus/\
   lm/${TRG}.blm:8 \
   -temp-dir ${MAIN_PATH}/translator_data_vn/tmp 2>&1
```

Rys. 6.11: Polecenie uruchomienia programu *train-model* do wytrenowania modelu tłumaczenia statystycznego.

```
${MOSES_BIN_DIR}/processPhraseTableMin -in $< \
   -out translator_data_vn/arena/model/phrase-table \
   -nscores 4 -threads ${NUM_THREADS}
```

Rys. 6.12: Polecenie uruchomienia programu *processPhraseTableMin* do stworzenia modelu tłumaczenia w wersji binarnej.

```
perl -i -pne \
   's/PhraseDictionaryMemory/PhraseDictionaryCompact$1/' $@
perl -i -pne \
   's/path=(.*phrase-table)\.gz/path=\.\/translator_data_vn\/arena
   \/model\/phrase-table.minphr/' $@
```

Rys. 6.13: Polecenia, które przełączają uruchomienie modelu tłumaczenia na jego wersję binarną.

Dla tłumaczenia neuronowego można byłoby już przekazać przygotowany tekst do wytrenowania modelu tłumaczenia neuronowego, ale można polepszyć wynik tłumaczenia neuronowego przez zastosowanie podziału słów na jednostki podwyrazowe. W tym celu w projekcie skorzystano z narzędzia *subword-nmt*. Najpierw wyuczono narzędzie jednostek podwyrazowych (Rys.6.17) przez zawarty w nim program *learn_bpe* (Rys.6.14). Następnie za pomocą programu *apply_bpe.py* (Rys.6.15) z narzędzia *subword-nmt* segmentuje się słowa z korpusu języka angielskiego i korpusu języka polskiego w celu przygotowania danych wejściowych do tłumaczenia neuronowego (Rys. 6.18). Na koniec stosuje się program w systemie Marian o nazwie *marian* (Rys.6.16), który trenuje model tłumaczenia neuronowego powieści wizualnych.

```
cat $^ | $(BPE_DIR)/learn_bpe.py -s $(MAX_WORDS) > $@
```

Rys. 6.14: Polecenie uruchomienia programu *learn_bpe* do wyuczenia jednostek podwyrazowych.

```
$(BPE_DIR)/apply_bpe.py -c translator_data_all_NMT_5/model/${SRC}${
TRG}.bpe < $< > $@
```

Rys. 6.15: Polecenie uruchomienia programu *apply_bpe.py* do przygotowania korpusu z jednostkami podwyrazowymi.

```
$(MARIAN_DIR)/marian --type s2s -e 5 --train-set $^ \
   --model $@  --disp-freq 100
```

Rys. 6.16: Polecenie uruchomienia programu *marian* do stworzenia modelu tłumaczenia neuronowego.

```
39951 wierzy my</w>
39952 wiel kość</w>
39953 wiecz ny</w>
39954 widy wal
39955 wia dają</w>
39956 wea ker</w>
39957 waż niejszy</w>
39958 wariat em</w>
39959 war ko
39960 waka cji</w>
```

Rys. 6.17: Przykłady wyuczonych jednostek podwyrazowych.

```
10 Beyond the reception desk , inside the ther@@ a@@ pi@@ st
   &apos; s room , I sat sitting face to face with the attending
   physician .
11 The room itself was a spa@@ cious 500 square feet or more . A
   comfortable sofa and a sur@@ real land@@ sca@@ pe painting
   are present to relieve anxiety , and a pair of ma@@ tching
   tea@@ -@@ colored book@@ cases position themselves alongside
   the daily com@@ mo@@ di@@ ties .
12 Even the light music that was circu@@ lating the room was
   there to target the sou@@ ght-@@ after goal of soo@@ thing a
   patient &apos; s soul .
```

Rys. 6.18: Korpus języka angielskiego po segmentacji słów.

## 6.4 Tłumaczenie i ocenianie

Zarówno w tłumaczeniu statystycznym, jak i w tłumaczeniu neuronowym, przed tłumaczeniem tekstu wymaga się przygotowania go przez tokenizację. W tym celu użyto programu zawierającego się w systemie Moses o nazwie *tokenizer* (Rys.6.6). Dodatkowo, przy tłumaczeniu neuronowym, w którym zastosowano w danych treningowych podział na jednostki podwyrazowe, wymagana jest segmentacja tekstu na jednostki podwyrazowe. Do tego użyto skryptu w języku Python *apply_bpe* zawartego w narzędziu *subword-nmt*.

Po przygotowaniu tekstu powieści wizualnych w języku angielskim następuje tłumaczenie powieści wizualnych na język polski. Dla tłumaczenia statystycznego użyto programu z systemu Moses o nazwie *moses* (Rys.6.19), a w tłumaczeniu neuronowym skorzystano z programu z systemu Marian o nazwie *marian-decoder* (Rys.6.21).

Po przetłumaczeniu tekstu powieści wizualnej trzeba poprawić znaki interpunkcyjne, które zostały zamienione przez działanie programu *tokenizer* na specjalne oznaczenia tych znaków w systemie Moses. Służy do tego program *deescape-special-chars* (Rys.6.20) z systemu Moses, który zmienia specjalne oznaczenia znaków interpunkcyjnych systemu Moses na ich zwykłe odpowiedniki. Dodatkowo przy tłumaczeniu neuronowym przed zastosowaniem programu *deescape-special-chars* należy usunąć podział

na jednostki podwyrazowe. W tym celu skorzystano z polecenia systemu Linux *sed* (Rys.6.21).

```
./Script/preprocess.sh ${SRC} < $< |\
   ${MOSES_BIN_DIR}/moses \
   -f translator_data_vn/arena/model/moses.ini \
   -search-algorithm 1 -threads ${NUM_THREADS} \
   | ./Script/postprocess.sh ${TRG} > $@
```

Rys. 6.19: Sekwencja poleceń do tłumaczenia statystycznego.

```
#!/bin/bash

LANG=$1

: ${MOSES_SCRIPTS_DIR:=/home/deinonzch/Pulpit/mosesdecoder/\
   scripts}

${MOSES_SCRIPTS_DIR}/tokenizer/deescape-special-chars.perl
```

Rys. 6.20: Skrypt bashowy uruchamiający program *deescape-special-chars*, który zamienia oznaczenia interpunkcyjne systemu Moses na zwykłe znaki interpunkcyjne.

```
./Script/preprocess.sh ${SRC} < $< \
   | $(BPE_DIR)/apply_bpe.py \
   -c translator_data_all_NMT/model/${SRC}${TRG}.bpe \
   | $(MARIAN_DIR)/marian-decoder \
   -m translator_data_all_NMT/model/model.npz --type s2s \
   -v translator_data_all_NMT/corpus/\
   preprocessed.bpe.${SRC}.yml \
   translator_data_all_NMT/corpus/\
   preprocessed.bpe.${TRG}.yml \
   | sed -r 's/(@@ )|(@@ ?$$)//g' \
   | ./Script/postprocess.sh ${TRG} > $@
```

Rys. 6.21: Sekwencja poleceń do tłumaczenia neuronowego.

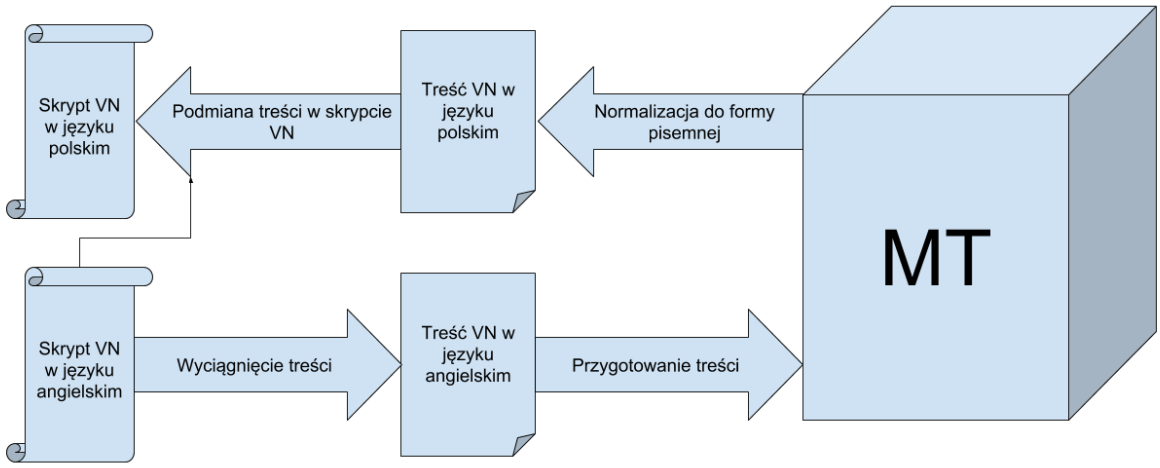
Do ocenienia przetłumaczonego tekstu zastosowano program *multi-bleu* (Rys.6.22) z systemu Moses, który ocenia wykonane tłumaczenia za pomocą metryki *BLEU*. Do poprawnego ocenienia przetłumaczonego tekstu wymagane jest, aby referencyjny korpus tekstowy również był stokenizowany (ponieważ przetłumaczony tekst jest stokenizowny po przetłumaczeniu) w tym celu zastosowano programy z systemu Moses *tokenizer* i *deescape-special-chars*.

```
1 ${MOSES_SCRIPTS_DIR}/generic/multi-bleu.perl $(word 2,$^) \
2     < $< > $@
```

Rys. 6.22: Polecenie do ocenienia tłumaczenia przy pomocy metryki BLEU.

## 6.5 System do tłumaczenia skryptów powieści wizualnych

W ramach projektu został stworzony autorski system o nazwie *Visual_Novel_Translator* do tłumaczenia powieści wizualnych (Rys.6.23) z języka angielskiego na język polski, który tłumaczy powieści oparte na jednym z trzech silników do gier powieści wizualnej (KiriKiri, Ren'Py, NScripter). System ten jest do pobrania na stronie *https://github.com/Deinonzch/Visual_Novel_Translator*. Do działania systemu wymagane jest zainstalowanie systemu Moses, systemu Marian, narzędzia Hunalign i narzędzia *subword-nmt* na komputerze z systemem operacyjnym Linux z kartą NVIDIA, która obsługuje architekturę CUDA.




Rys. 6.23: Rysunek przedstawiający działanie translatora powieści wizualnych.

W celu stworzenia modeli tłumaczenia powieści wizualnych z języka angielskiego na polski wystarczy podać polecenie *make* w konsoli systemowej i wcześniej w pliku *Makefile* podstawić odpowiednio ścieżki dostępu do powyższych narzędzi. Polecenie do uruchomienia programu tłumaczącego ma postać:

*./translator.sh [silnik powieści wizualnej] [ścieżka dostępu do skryptu z treścią powieści wizualnej] [ścieżka dostępu do przetłumaczonego skryptu] [model tłumaczenia].*

Jako wartość opcji [silnik powieści wizualnej] trzeba podać silnik gry powieści wizualnej, przy czym dla silnika KiriKiri ta wartość wynosi *kirikiri*, dla silnika NScripter to *nscripter*, a dla silnika Ren'Py to *renpy*. Dwa następne argumenty to odpowiednio ścieżka dostępu do skryptu i ścieżka dostępu do zapisania przetłumaczonego skryptu. Ostatni argument odpowiada za model tłumaczenia. Domyślnie przyjmuje się model tłumaczenia statystycznego wyuczonego na zbiorze danych powieści wizualnych i napisów filmowych. Do podania innego modelu tłumaczenia trzeba wpisać wartość *my_model*, wówczas po zatwierdzeniu polecenia użytkownik zostanie poproszony o podanie ścieżki dostępu do innego modelu tłumaczenia i podania rodzaju modelu (smt - model tłumaczenia statystycznego, nmt - model tłumaczenia neuronowego)(Rys.6.24).

```
1 ./translator.sh nscripter Data/scripts_VN/onik_000.txt translator/
  translated_script/onik_000.txt my_model
2 Provide the access path to the model, for example:
  translator_data_all_NMT
3 translator_data_all_NMT
4 Provide the kind of model(please write smt or nmt)
5 nmt
```

Rys. 6.24: Przykład działania polecenia *translator*.

Pierwszym krokiem programu tłumaczącego jest wyciągnięcie treści powieści wizualnej z pliku gry powieści wizualnej. Służą do tego autorskie skrypty w języku Python (Rys.6.25), które wyciągają treść powieści wizualnej spod konkretnego pliku z silnika gry powieści wizualnej.

```
1 #!/usr/bin/python3
2 # -*- coding: utf-8 -*-
3 import re
4 import sys
5 import argparse
6
7 parser = argparse.ArgumentParser()
```

```
parser.add_argument("text_file",  type= str,
                    help="text from script")
parser.add_argument("quotation_file",  type= str,
                    help="quotation from script")

args = parser.parse_args()

regex = re. compile(r"(\")(.*)(\")")

data_english = []
data_quotation = []

i = 1
for line in sys.stdin:
    _str = re.findall(regex, line)
    if _str != []:
        if _str[0] != "":
            if i == 1:
                data_english.append(_str[0][1])
                data_quotation.append(_str[0][0] + ' '
                                      + _str[0][2])
                i = 2
            else:
                i = 1
with  open(args.text_file, 'w') as tsvfile:
    for each in data_english:
        tsvfile.write(each + "\n")

with  open(args.quotation_file, 'w') as tsvfile:
    for each in data_quotation:
        tsvfile.write(each + "\n")
```

Rys. 6.25: Skrypt do wyciągania tekstu z pliku silnika Ren'Py zawierającego powieść wizualną.

Następnym krokiem programu do tłumaczenia powieści wizualnej jest przygotowanie tekstu do tłumaczenia i jego przetłumaczenie, które zostało omówione w rozdziale 7.4. Po tym następuje krok norma-

lizacji przetłumaczonego tekstu do formy pisemnej. W tym celu najpierw trzeba przywrócić normalne oznaczenia interpunkcyjne za pomocą programu systemu Moses *deescape-special-chars* (przy tłumaczeniu neuronowym trzeba jeszcze przed tym usunąć podział na jednostki podwyrazowe na przykład za pomocą polecenia systemu Linux *sed*). Drugim krokiem normalizacji przetłumaczonego tekstu do formy pisemnej jest jego właściwe doprowadzenie do formy pisemnej, które realizuje autorski skrypt w języku Python *correct_output* (Rys.6.26), który sprowadza stokenizowany teksty do formy pisemnej.

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-
import sys
import re

regex = re. compile(r'([A-Za-zEÓŁŚĄŹŻĆŃéóĺśáźžćń0-9-])
                    [A-Za-zEÓŁŚĄŹŻĆŃéóĺśáźžćń0-9-]*')
#znak jest za znakiem
regex_bad_interpuction_1 = re. compile(r'([A-Za-zEÓŁŚĄŹŻĆŃ
                                          éóĺśáźžćń0-9-\?\!
                                          \.,;])([\(\’\´\"])')
#po znaku jest spacja
regex_bad_interpuction_2 = re. compile(r'\s([\(\’\´\"])(\s)')

for line in sys.stdin:
    tab = line.split(' ')
    string = ''
    i = 0
    for each in tab:
        i = i + 1
        if each == '':
          string = string + ' ' + each
        else:
          if re.findall(regex, each):
              if string == '"' or string == '´'
                      or string == '’':
                  string = string + each
              else:
```

```
29                  if string == '"\\"':
30                      string = string + each
31                  else:
32                      if i == 1:
33                          string = each
34                      else:
35                          string = string + ' ' + each
36                      #print(string)
37          else:
38              string = string + each
39              #print(string)
40          #print(each)
41          if each == '\n':
42              i = 0
43      string = string.replace('  ', ' ')
44      bad_interpuction_1 = re.findall(regex_bad_interpuction_1,
45                                      string)
46      for each in bad_interpuction_1:
47        string = string.replace(each[0]+each[1], each[0]
48                                + ' ' + each[1])
49      bad_interpuction_2 = re.findall(regex_bad_interpuction_2,
50                                      string)
51      for each in bad_interpuction_2:
52        string = string.replace(each[0]+each[1], each[0])
53      if string[0] == ' ' or string[0] == '´':
54        #myślnik na początku
55        if string[1:4] == '- ':
56          string = string[0:3] + string[4:]
57        #trójkropek na początku
58        if string[1:5] == '... ':
59          string = string[0:4] + string[5:]
60      else:
61        #myślnik na początku
62        if string[0:3] == '- ':
```

```
63          string = string[0:2] + string[3:]
64        #trójkropek na początku
65        if string[0:4] == '... ':
66          string = string[0:3] + string[4:]
67
68      print(string[:-1])
```

Rys. 6.26: Skrypt w języku Python, który sprowadza stokenizowany tekst do formy pisemnej.

Ostatnim krokiem tłumaczenia powieści wizualnej jest podmiana zawartości skryptu powieści wizualnej w języku angielskim na przetłumaczoną treść w języku polskim. Służą do tego autorskie skrypty w języku Python (Rys.6.27), które podmieniają treść angielską powieści wizualnej na przetłumaczoną treść w języku polskim pod konkretny silnik gry powieści wizualnej.

```
1  #!/usr/bin/python3
2  # -*- coding: utf-8 -*-
3  import re
4  import argparse
5
6  parser = argparse.ArgumentParser()
7  parser.add_argument("script_file",  type= str,
8                      help="renpy script")
9  parser.add_argument("translated_text",  type= str,
10                     help="translated text from renpy script")
11 parser.add_argument("quotation_file",  type= str,
12                     help="quotation from script")
13
14
15 args = parser.parse_args()
16
17 regex_renpy = re. compile(r'script\.(rpy)')
18 regex_path = re. compile(r'((.+\/)*(.+\..+))')
19 regex_text = re. compile(r"(\".*\")")
20 regex_quotation = re. compile(r"(\").*(\")")
21
22 if re.findall(regex_renpy, args.script_file)
```

```
        and re.findall(regex_path, args.script_file)[0][0]
        == args.script_file
        and re.findall(regex_path, args.translated_text)[0][0]
        == args.translated_text:
    with  open(args.script_file, 'r') as script:
        with  open(args.translated_text, 'r') as text:
            with  open(args.quotation_file, 'r') as quotation:
                i = 1
                for line in script:
                    _str = re.findall(regex_text, line)
                    if _str != []:
                        if _str[0] != "":
                            if i == 1:
                                print(line[:-1])
                                i = 2
                            else:
                                quot = re.findall(
                                    regex_quotation,
                                    quotation.readline())
                                print(line.replace('"',
                                                   quot[0][0]
                                                   + text.
                                                   readline()
                                                   [:-1]
                                                   + quot
                                                   [0][1])
                                      [:-1])
                                i = 1
                    else:
                        print(line[:-1])
else:
    print('It is not script.rpy or one of '
          'two arguments is not path')
```

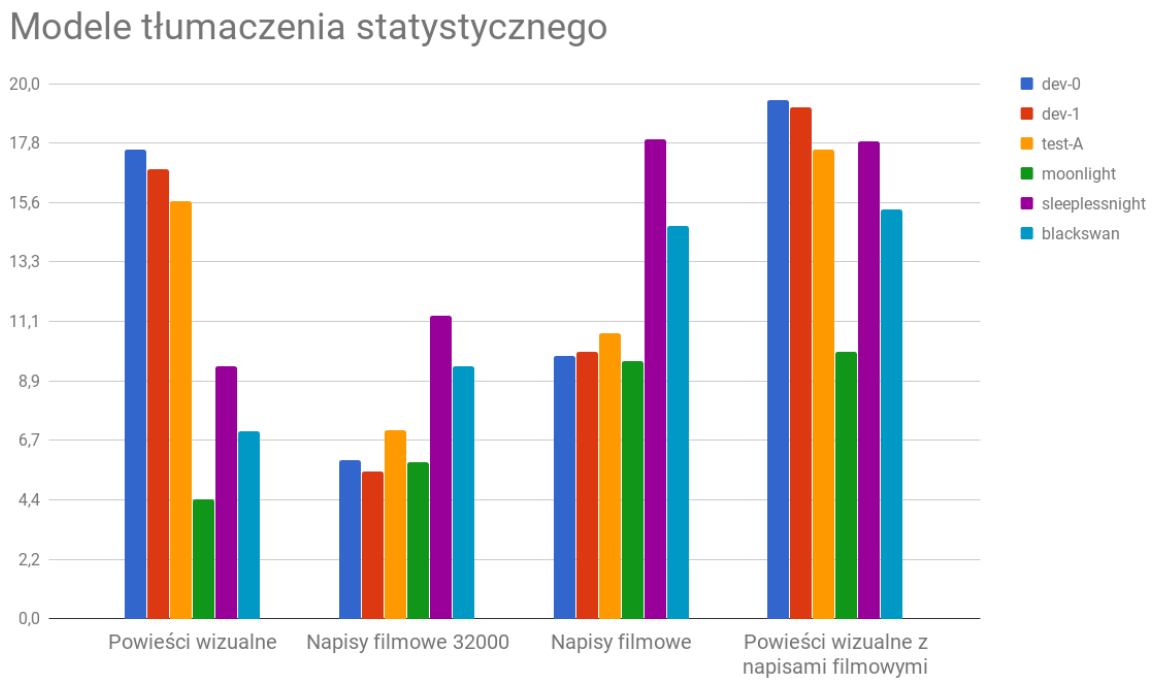
Rys. 6.27: Skrypt do podmiany tekstu w języku angielskim z pliku silnika Ren'Py na jego przetłumaczony odpowiednik w języku polskim.

## 6.6 Tłumaczenie statystyczne vs neuronowe

Powieści wizualne użyte w projekcie to *G-senjou no Maou* (powieść wizualna gatunku kryminał, dreszczowiec, erotyczna i romans), *To Kill A Black Swan* (powieść wizualna gatunku kryminał), *Sleepless Night* (powieść gatunku horror), *Moonlight Walks* (powieść gatunku romans).

Do wytrenowania modeli tłumaczenia i stworzenia zbiorów deweloperskich i testowego użyto powieść wizualną *G-senjou no Maou*, która w wersji angielskiej liczy 41266 wierszy tekstu i 41265 wierszy tekstu w wersji polskiej. Zbiory deweloperskie *dev-0* i *dev-1* liczą odpowiednio 3191 wierszy i 3071 wierszy. Zbiór testowy z powieści wizualnej *G-senjou no Maou* liczy 3293 wierszy.

Pozostała część korpusu z powieści wizualnej *G-senjou no Maou* została przeznaczona do wytrenowania modeli tłumaczenia. Pozostałe powieści (*To Kill A Black Swan*, *Sleepless Night*, *Moonlight Walks*, które liczą odpowiednio 940, 292, 546 wierszy tekstu) służą jako osobne zbiory testowe. Do zbioru treningowego należą także dwa korpusy napisów filmowych liczących po milion zdań.




Rys. 6.28: Wyniki metryki BLEU dla tłumaczenia statystycznego.

W legendzie Rys. 6.28 *dev-0*, *dev-1* i *test-A* są zbiorami z powieści wizualnej *G-senjou no Maou*, gdzie dwa pierwsze to zbiory deweloperskie, a ostatni to zbiór testowy. Kolejne trzy kategorie to *mo-*

*onlight*, *sleeplessnight* i *blackswan*, które odpowiednio odpowiadają powieściom wizualnym *Moonlight Walks*, *Sleepless Night* i *To Kill A Black Swan*. Słupki zostały podzielone na cztery grupy, które przedstawiają modele tłumaczenia statystycznego wytrenowane na zbiorach:

1. *Powieści wizualne* - model wytrenowany na zbiorze trenującym składającym się tylko z powieści wizualnej *G-senjou no Maou* liczącym około 32000 zdań

2. *Napisy filmowe 32000* - model wytrenowany na zbiorze trenującym składającym się tylko z napisów filmowych liczącym 32000 zdań

3. *Napisy filmowe* - model wytrenowany na zbiorze trenującym składającym się tylko z napisów filmowych liczącym milion zdań

4. *Powieści wizualne z napisami filmowymi* - zbiór trenujący łączący zbiory *Powieści wizualne* i *Napisy filmowe*

Wyniki oceny miary BLEU dla statystycznego tłumaczenia automatycznego dla powieści wizualnych (Rys.6.28) przedstawiają się następująco:

- Ocena BLEU dla zbiorów deweloperskich i testowego z *G-senjou no Maou* ma najwyższe wyniki w modelach opartych na zbiorze treningowym z tej samej powieści wizualnej.

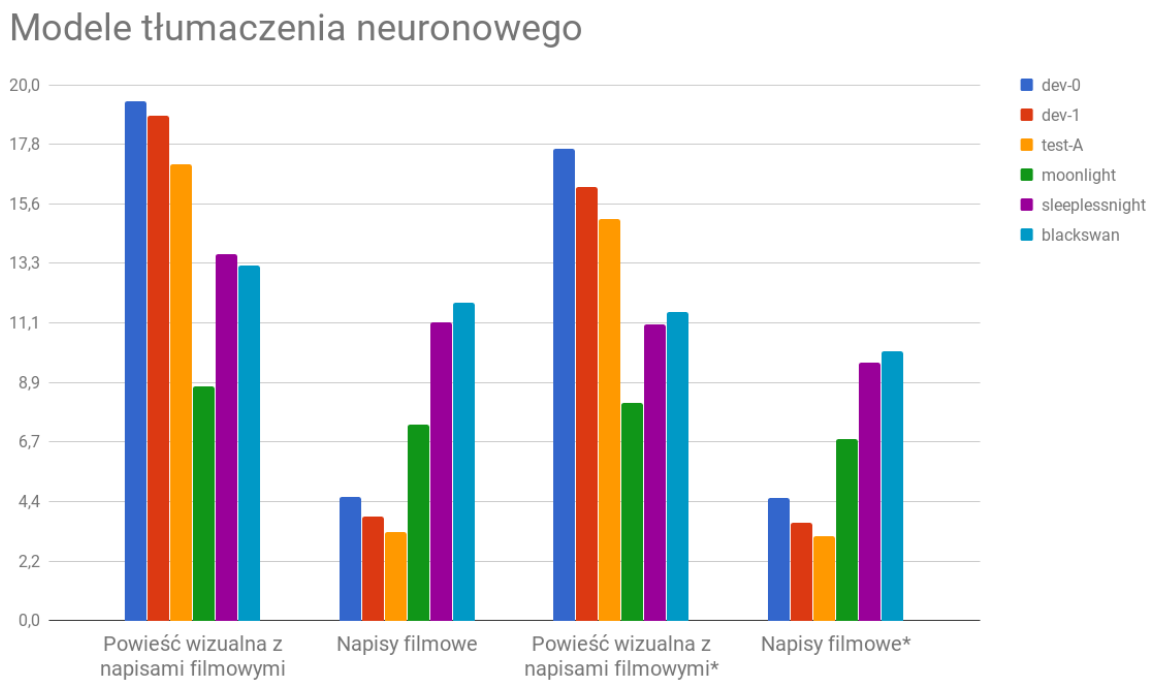
- Ocena BLEU dla *To Kill A Black Swan*, *Sleepless Night*, *Moonlight Walks* dla modelu wytrenowanego z napisów filmowych liczących 32000 zdań jest o około dwa punkty BLEU wyższa od modelu wytrenowanego na powieści *G-senjou no Maou* (około 32000 wierszy).

- Ocena BLEU danych treningowych dla modelu wytrenowanego na milionie zdań z napisów filmowych jest o co najmniej kilka punktów BLEU wyższa od modelu wytrenowanego na 32000 zdań z napisów filmowych.

- Ocena BLEU *Sleepless Night* w modelu wytrenowanym na milionie zdań z napisów filmowych jest porównywalny z oceną zbiorów deweloperskich i testowego z *G-senjou no Maou* w modelu wytrenowanym z powieści wizualnej *G-senjou no Maou*.

- Ocena BLEU wszystkich zbiorów treningowych wzrosła w modelu wytrenowanego na zbiorze powieści wizualnych z milionem wierszy napisów filmowych w stosunku do modelu wytrenowanego na zbiorze powieści wizualnych. Było to około 2 punktów BLEU dla zbiorów *G-senjou no Maou*, około 5 punktów BLEU dla *Moonlight Walks* i około 8 punktów BLEU dla *To Kill A Black Swan* i *Sleepless Night*.

- Ocena BLEU prawie wszystkich zbiorów treningowych wzrosła w modelu wytrenowanego na zbiorze powieści wizualnych z milionem wierszy napisów filmowych w stosunku do modelu wytrenowanego na milionie wierszy napisów filmowych. Było to około 10, 9 i 7 punktów BLEU odpowiednio dla zbirów *dev-0*, *dev-1* i *test-A* i niewielka zmiana punktów BLEU dla zbiorów *To Kill A Black Swan* i *Moonlight Walks* odpowiednio o około 0,6 i 0,3 punkta BLEU oraz bardzo mały spadek dla *Sleepless Night* o około 0,2 punkta BLEU.




Rys. 6.29: Wyniki metryki Blue dla tłumaczenia neuronowego.

Zgodnie z postulatami zawartymi w pracach [SHB16] i [LPM15] został użyty model tłumaczenie neuronowego oparty na mechanizmie atencji i na jednostkach podwyrazowych. Sensowne wyniki badań można było dopiero uzyskać po dostarczeniu korpusu liczącego ponad milion wierszy. Dlatego wyniki badań nie biorą pod uwagę zbioru treningowego liczącego tylko powieści wizualne (około 32000 wierszy tekstu) i zbioru z napisów filmowych liczących tylko 32000 zdań.

W legendzie Rys. 6.29 *dev-0*, *dev-1* i *test-A* są zbiorami z powieści wizualnej *G-senjou no Maou*, gdzie dwa pierwsze to zbiory deweloperskie, a ostatni to zbiór testowy. Kolejne trzy kategorie to *moonlight*, *sleeplessnight* i *blackswan*, które odpowiednio odpowiadają powieściom wizualnym *Moonlight Walks*, *Sleepless Night* i *To Kill A Black Swan*. Słupki zostały podzielone na cztery grupy, które przedstawiają modele tłumaczenia neuronowego wytrenowane na zbiorach:

1. *Powieści wizualne z napisami filmowymi* - model wytrenowany na 10 powtórzeniach na zbiorze

trenującym składającym się z około 32000 zdań powieści wizualnej *G-senjou no Maou* i z miliona zdań napisów filmowych

2. *Napisy filmowe* - model wytrenowany na 10 powtórzeniach na zbiorze trenującym składającym się z miliona zdań napisów filmowych

3. *Powieści wizualne z napisami filmowymi** - model wytrenowany na 5 powtórzeniach na zbiorze trenującym składającym się z około 32000 zdań powieści wizualnej *G-senjou no Maou* i z miliona zdań napisów filmowych

4. *Napisy filmowe** - model wytrenowany na 5 powtórzeniach na zbiorze trenującym składającym się z miliona zdań napisów filmowych

Wyniki miary BLEU dla neuronowego tłumaczenia automatycznego powieści wizualnych (Rys.6.29) są następujące:

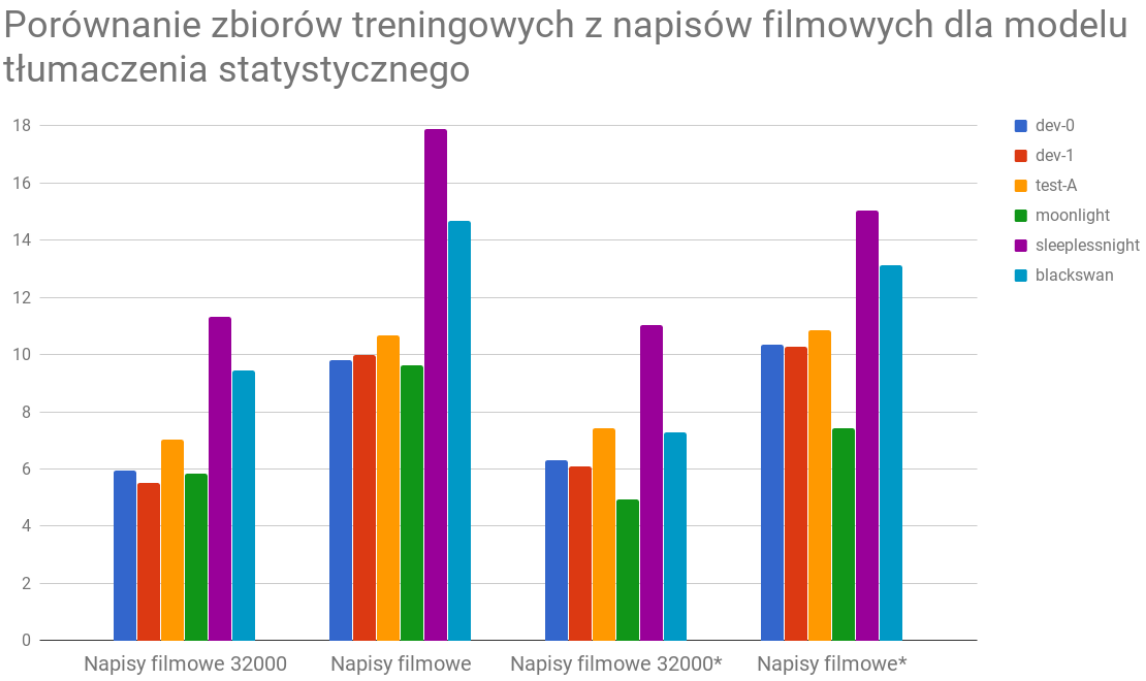
- Nie ma dużej różnicy miary BLEU dla zbiorów z powieści wizualnej *G-senjou no Maou* przy modelu wytrenowanym na napisach filmowych liczących milion zdań przy 10 i 5 powtórzeniach.

- Dla modelu wytrenowanego na napisach filmowych (milion zdań) przy 10 powtórzeniach miara BLEU jest o około 2 punkty BLEU wyższa dla zbiorów treningowych *To Kill A Black Swan*, *Sleepless Night* i o 0,5 punktu BLEU wyższa dla zbioru treningowego *Moonlight Walks* w stosunku do tego samego modelu, ale wytrenowanego na 5 powtórzeniach.

- Dla modelu wytrenowanego na powieściach wizualnych z napisami filmowymi przy 10 powtórzeniach jest widoczna różnica oceny BLEU dla zbiorów testowych i deweloperskich w stosunku do tego samego modelu, ale wytrenowanego na 5 powtórzeniach.

- Dla modelu wytrenowanego na powieściach wizualnych z napisami filmowymi przy 10 powtórzeniach ocena BLEU dla zbioru *Sleepless Night* jest wyższa od zbioru *To Kill A Black Swan* w stosunku do modelu wytrenowanego na napisach filmowych przy 10 powtórzeniach.

Wysokie wyniki dla powieści wizualnych *To Kill A Black Swan*, *Sleepless Night*, *Moonlight Walks* przetłumaczonych w modelach wytrenowanych na napisach filmowych, skłoniły do wytrenowania drugiego modelu tłumaczenia statystycznego opartego na innym korpusie napisów filmowych z *OpenSubtitles2018* w celu porównania wyników między nimi.




Rys. 6.30: Wyniki metryki BLEU dla tłumaczenia statystycznego na zbiorach treningowych z napisów filmowcyh.

W legendzie Rys. 6.30 *dev-0*, *dev-1* i *test-A* są zbiorami z powieści wizualnej *G-senjou no Maou*, gdzie dwa pierwsze to zbiory deweloperskie, a ostatni to zbiór testowy. Kolejne trzy kategorie to *moonlight*, *sleeplessnight* i *blackswan*, które odpowiednio odpowiadają powieściom wizualnym *Moonlight Walks*, *Sleepless Night* i *To Kill A Black Swan*. Słupki zostały podzielone na cztery grupy, które przedstawiają modele tłumaczenia statystycznego wytrenowane na zbiorach:

1. *Napisy filmowe 32000* - pierwszy model wytrenowany na zbiorze trenującym składającym się tylko z napisów filmowych liczącym 32000 zdań

2. *Napisy filmowe* - pierwszy model wytrenowany na zbiorze trenującym składającym się tylko z napisów filmowych liczącym milion zdań

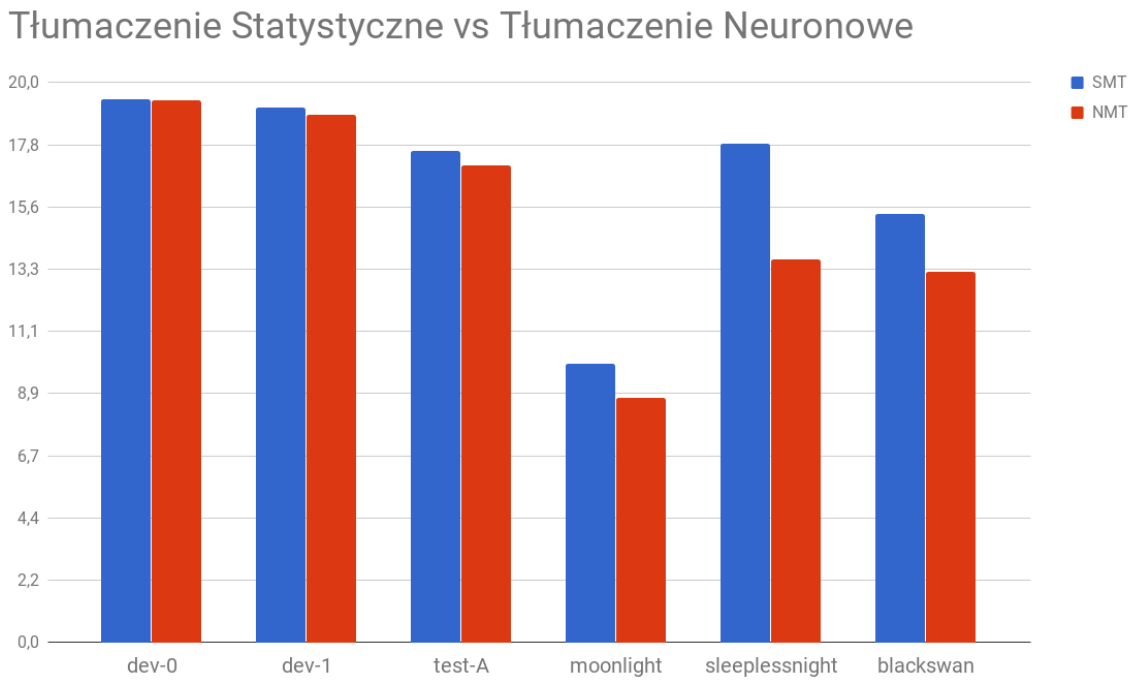
3. *Napisy filmowe 32000** - drugi model wytrenowany na zbiorze trenującym składającym się tylko z napisów filmowych liczącym 32000 zdań

4. *Napisy filmowe** - drugi model wytrenowany na zbiorze trenującym składającym się tylko z napisów filmowych liczącym milion zdań

Porównanie miary BLEU między modelami wytrenowanymi na pierwszym i drugim korpusie z napisów filmowych (Rys.6.30) przedstawia się następująco:

- Ocena BLEU dla modelu wytrenowanego na drugim korpusie (oznaczony * na Rys.6.30) liczącym 32000 zdań napisów filmowych jest o około 0,5 punktu BLEU wyższa dla zbiorów *G-senjou no Maou* i gorsza dla zbiorów *To Kill A Black Swan*, *Sleepless Night*, *Moonlight Walks* odpowiednio o około 2, 0,5 i 1 punktu BLEU w stosunku do modelu wytrenowanego na pierwszym korpusie liczącym 32000 napisów filmowych.

- Ocena BLEU dla modelu wytrenowanego na drugim korpusie (oznaczony * na Rys.6.30) liczącym milion zdań napisów filmowych jest o około 0,5 punktu BLEU wyższa dla zbiorów *G-senjou no Maou* i niższa dla zbiorów *To Kill A Black Swan*, *Sleepless Night*, *Moonlight Walks* odpowiednio o około 1, 3 i 2 punktów BLEU w stosunku do modelu wytrenowanego na pierwszym korpusie liczącym milion napisów filmowych.




Rys. 6.31: Wyniki metryki BLEU dla tłumaczenia statystycznego i neuronowego.

W legendzie Rys. 6.31 *SMT* i *NMT* oznaczają odpowiednio modele tłumaczenia statystycznego i tłumaczenia neuronowego. Obydwa modele zostały wytrenowane na zbiorze trenującym składającym się z około 32000 zdań z powieści wizualnej *G-senjou no Maou* i milionie zdań z napisów filmowych. Słupki zostały podzielone na sześć grup, które przedstawiają zbiory deweloperskie i testowe:

1. *dev-0* - zbiór deweloperski składający się z powieści wizualnej *G-senjou no Maou*

2. *dev-1* - zbiór deweloperski składający się z powieści wizualnej *G-senjou no Maou*

3. *test-A* - zbiór testowy składający się z powieści wizualnej *G-senjou no Maou*

4. *moonlight* - zbiór testowy składający się z powieści wizualnej *Moonlight Walks*

5. *sleeplessnight* - zbiór testowy składający się z powieści wizualnej *Sleepless Night*

6. *blackswan* - zbiór testowy składający się z powieści wizualnej *To Kill A Black Swan*

Wyniki miary BLEU dla zbiorów testowych i deweloperskich między modelami tłumaczenia statystycznego i tłumaczenia neuronowego wytrenowanymi na zbiorze trenującym składającym się z powieści wizualnej *G-senjou no Maou* i z napisów filmowych (Rys.6.31) przedstawiają się następująco:

- Ocena BLEU dla modeli tłumaczenia statystycznego jest lepsza od oceny BLEU dla modeli tłumaczenia neuronowego dla zbiorów testowych *moonlight*, *sleeplessnight*, *blackswan*

- Ocena BLEU dla modeli tłumaczenia statystycznego jest porównywalna do oceny BLEU dla modeli tłumaczenia neuronowego dla zbiorów składających się z powieści wizualnej *G-senjou no Maou*

- Największa różnica miary BLEU między tłumaczeniem statystycznym a neuronowym jest dla *sleeplessnight*, a najmniejsza różnica jest dla *dev-0*

- Wyniki miary BLEU są lepsze dla tłumaczenia statystycznego niż dla tłumaczenia neuronowego o 0,02 punktu BLEU dla *dev-0*, o 0,27 punktu BLEU dla *dev-1*, o 0,52 punktu BLEU dla *test-A*, o 1,22 punktów BLEU dla *moonlight*, o 4,15 punktów BLEU dla *sleeplessnight* i o 2,06 punktów BLEU dla *blackswan*

# Rozdział 7

# Podsumowanie

Celem projektu zaprezentowanego w niniejszej pracy było stworzenie systemu tłumaczącego powieści wizualne z języka angielskiego na polski i porównanie tłumaczenia statystycznego z tłumaczeniem neuronowym.

Udało się stworzyć system tłumaczenia powieści wizualnych, który tłumaczy powieści wizualne co najwyżej w niskiej jakości. Nie udało się przeprowadzić porównania tłumaczenia statystycznego z tłumaczeniem neuronowym dla olbrzymich korpusów składających się z kilku milionów zdań z powieści wizualnych, ponieważ nie udało się zdobyć wystarczającej ilości danych z powodu zbyt małej ilości powieści wizualnych, które są zarówno w języku angielskim, jak i w polskim. Udało się porównać tłumaczenie statystyczne i neuronowe dla zbiorów treningowych składających się z około 32000 zdań z powieści wizualnej.

Według pracy [WM15] wyniki miary BLEU pokazują, że system tłumaczenia powieści wizualnych jest w stanie zapewnić wysoką jakość tłumaczenia niektórych powieści. Z powodu podobieństw powieści wizualnych i napisów filmowych istnieje możliwość, że wyniki pracy [VSHT10] da się przełożyć na powieści wizualne. Zatem istnieje możliwość uzyskania o wiele wyższej oceny BLEU, gdy zostanie użyty duży, wysokiej jakości korpus równoległy. Zatem dalsze prace powinny się z pewnością skupić na lepszym przygotowaniu danych wejściowych.

Tłumaczenie statystyczne lepiej sobie radzi z tłumaczeniem na mniejszej ilości danych od tłumaczenia neuronowego, jest zgodna z wynikami w pracy [KK17].

Z powodu otrzymania wyższych wyników miary BLEU w tłumaczeniu zbiorów *G-senjou no Maou* przy modelach wytrenowanych na zbiorze treningowym zawierającą tę samą powieść, planuje się sprawdzić tłumaczenie kolejnych części powieści za pomocą poprzednich części tytułu.

Skrypty wydobywające treść z powieści wizualnej mogą się przydać tłumaczom, którzy tłumaczą powieści wizualne. Skrypty te można wykorzystać do skonstruowania aplikacji do wyciągania treści z powieści wizualnych.

# Bibliografia


[JD08] Marcin Junczys-Dowmunt. Wprowadzenie do metod statystycznych w tłumaczeniu automatycznym. 2008.

[KK17] Philipp Koehn and Rebecca Knowles. Six challenges for neural machine translation. http://www.aclweb.org/anthology/W17-3204, 2017.

[Koe10a] Philipp Koehn. *Statistical Machine Translation*, pages 181–214. Cambridge, 2010.

[Koe10b] Philipp Koehn. *Statistical Machine Translation*, pages 217–244. Cambridge, 2010.

[Koe17] Philipp Koehn. Statistical machine translation draft of chapter 13: Neural machine translation. https://arxiv.org/pdf/1709.07809.pdf, 2017.

[Koz14] Sebastian Kozłowski. Co to jest tłumaczenie maszynowe? http://web.archive.org/web/20140125011838/http://kf.mish.uw.edu.pl:80/kog/kog_seb.pdf, 2014. s. 68–71.

[LK11] Josiah Lebowitz and Chris Klug. *Interactive storytelling for video games: a player-centered approach to creating memorable characters and storie*, pages 192–193. Focal Press, 2011.

[LPM15] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. https://nlp.stanford.edu/pubs/emnlp15_attn.pdf, 2015.

[SHB16] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. http://www.aclweb.org/anthology/P16-1162, 2016.

[VSHT10] Martin Volk, Rico Sennrich, Christian Hardmeier, and Frida Tidström. Machine translation of tv subtitles for large scale production. http://www.zora.uzh.ch/id/eprint/36755/1/Volk_Sennrich_Hardmeier_Tidstr%C3%B6mV.pdf, 2010.

[WM15] Krzysztof Wołk and Krzysztof Marasek. Neural-based machine translation for medical text domain. based on european medicines agency leaflet texts. https://arxiv.org/pdf/1509.08644.pdf, 2015.

# Spis rysunków