Uniwersytet im. Adama Mickiewicza w Poznaniu

Wydział Matematyki i Informatyki

Tomasz Ziętkiewicz

Zaprojektowanie oraz implementacja systemu automatycznej korekcji błędów i normalizacji wyjścia z systemu rozpoznawania mowy

Rozprawa doktorska

Promotor:

prof. UAM dr hab. Jacek Marciniak

Promotor pomocniczy:

dr Marek Kubis

Dyscyplina:

Informatyka

Poznań 2023

Spis treści

Podziękowania

Składam serdeczne podziękowania promotorom: profesorowi UAM dr hab. Jackowi Marciniakowi i dr Markowi Kubisowi, za pomoc okazaną przez cały czas trwania doktoratu, liczne uwagi i wskazówki oraz za godziny spędzone na czytaniu kolejnych wersji pracy.

Dziękuję również profesorowi Krzysztofowi Jassemowi za poczyniony wkład w rozpoczęcie na wydziale programu "Doktorat wdrożeniowy" i za pozytywną motywację przekazywaną podczas licznych rozmów.

Dziękuję firmie Samsung Research Poland a w szczególności doktor Bożenie Łuksasik, dr Mikołajowi Wypychowi oraz Marcinowi Sowańskiemu – za okazane wsparcie zarówno w sprawach formalnych jak i merytorycznych oraz zapewnienie warunków do prowadzenia badań.

Chciałbym również wyrazić wdzięczność mojej rodzinie: rodzicom, dzieciom a przede wszystkim mojej żonie: za wykazaną cierpliwość i wspieranie mnie w drodze do celu.

Rozdział 1

Wprowadzenie

1.1. Motywacja

Historia rozwoju systemów rozpoznawania mowy sięga niemal początków rozwoju komputerów. Za pierwszy programowalny komputer uznaje się ENIAC[1], uruchomiony w 1945 roku. Zaledwie 7 lat później skonstruowano pierwszą maszynę, która była w stanie rozpoznawać pojedyncze liczebniki[2].

Minęło ponad 70 lat i dzisiaj systemy rozpoznawania mowy są wszechobecne: znajdziemy je w telefonach, telewizorach, samochodach, a nawet w lodówkach. Są używane zarówno w urządzeniach konsumenckich jak i w zastosowaniach profesjonalnych takich jak infolinie, transkrypcja materiałów audio i wideo, sporządzanie dokumentacji medycznej czy nawet sterowanie wojskowymi samolotami odrzutowymi. Można więc z pewnością stwierdzić, że technologia rozpoznawania mowy osiągnęła swoją dojrzałość, nie jest już tylko technologiczną ciekawostką, z której korzysta jedynie garstka entuzjastów nowych technologii.

Tak spektakularny rozwój technologii rozpoznawania mowy nie byłby możliwy gdyby nie rozwój modeli uczenia maszynowego, a zwłaszcza głębokich sieci neuronowych. Jeszcze około 15 lat temu większość problemów sztucznej inteligencji, takich jak rozpoznawanie mowy, obrazów czy tłumaczenia maszynowego była rozwiązywana przez systemy, których stworzenie i utrzymywanie wymagało nie tylko specjalistycznej wiedzy dotyczącej danego problemu, ale również wymagającej pracy polegającej na tworzeniu reguł i czasochłonnej konstrukcji i doborze cech wykorzystywanych przez tworzone modele [76]. Wykonywanie wielu z tych zadań przez człowieka nie jest dziś konieczne – głębokie sieci neuronowe są w stanie samodzielnie nauczyć się istotnych dla danego problemu cech na podstawie wystarczającej ilości danych trenujących [19][76]. Powstałe przy ich użyciu modele rozpoznawania obrazów [68], mowy [48][94], rozumienia tekstu [19] czy tłumaczenia maszynowego [134] osiągają niespotykane wcześniej rezultaty przy niewielkim nakładzie ze strony ludzkich ekspertów [76]. Wraz z ich rozwojem popełniają coraz mniej błędów, nieraz osiągając poziom zbliżony lub przewyższający ludzkie możliwości, tak jak

[1] ENIAC (Electronic Numerical Integrator and Computer) [147])

[2] Urządzenie nazwane "Audrey", skonstruowane w Bell laboratories w USA potrafiło rozpoznawań pojedyncze cyfry wymawiane przez tę samą osobę [26]

w przypadku tłumaczenia maszynowego [6][45], odpowiadania na pytania [152] czy rozpoznawania mowy [151][131].

Nie oznacza to jednak, że problem rozpoznawania mowy można uznać za rozwiązany i próżno na tym polu szukać wyzwań. Dlaczego zatem mimo bliskich ideałowi wyników raportowanych w publikacjach opisujących modele rozpoznawania mowy wciąż nie porzuciliśmy klawiatur na rzecz mikrofonów? Do korzystania z głosu jako sposobu interakcji z urządzeniem często zniechęcają nas błędy popełniane przez system ASR [150]. Choć najlepsze modele zamiany mowy na tekst osiągają wyniki porównywalne z ludzkimi anotatorami [151][131], to jest tak tylko w przypadku niektórych zbiorów danych – w przypadku nagrań wypowiedzi w trudnych warunkach akustycznych, w języku niebędącym językiem ojczystym mówcy, czy dotyczących specjalistycznych tematów, ludzie wciąż radzą sobie lepiej od maszyn [128][118][8]. Wyniki zbliżające się do 100% WRR[3] są osiągane na dobrze znanych korpusach mowy, na których modele te są doskonalone. Gdy badane są na wypowiedziach pochodzących z innych domen wyniki ulegają znacznemu pogorszeniu [97][118].

Najczęściej stosowane i osiągające najlepsze wyniki modele głębokich sieci neuronowych to modele typu *black box* (czarna skrzynka) [83], co utrudnia zrozumienie przyczyn popełnianych przez nie błędów i ich selektywne naprawianie. Adaptacja wykorzystywanego modelu rozpoznawania mowy, czyli proces dostosowania modelu do nowych danych, w celu poprawienia popełnianych przez niego błędów jest czasochłonna i kosztowna [106], może doprowadzić do pogorszenia wyników na starych danych [92][34]), w końcu może też być niemożliwa w przypadku korzystania z zamkniętych modeli.

Dlatego w celu praktycznego wykorzystania modeli rozpoznawania mowy, w praktyce biznesowej łączy się je z modułem, który umożliwia precyzyjną korektę błędów popełnianych przez taki model. W języku angielskim taki moduł nazywa się "post-processor", co na język polski można przetłumaczyć jako "Moduł przetwarzania końcowego". Moduł taki może zawierać zestaw reguł, które naprawiają błędy modelu statystycznego i dostosowują format danych wyjściowych do formatu oczekiwanego na wyjściu całego systemu (dokonują jego normalizacji).

Reguły te mogą być tworzone ręcznie, na przykład przy wykorzystaniu gramatyk i wyrażeń regularnych [154][124][130]. Ich stworzenie i utrzymanie wymaga jednak żmudnej pracy wysoce wykwalifikowanych specjalistów łączących umiejętności techniczne z językowymi. Z tego względu od pewnego czasu proponuje się modele statystyczne realizujące te zadania. Mogą one stanowić uzupełnienie podejścia regułowego, minimalizując nakład pracy potrzebny na utrzymanie ręcznie tworzonych reguł korekty i normalizacji.

Badania opisane w tej pracy były realizowane we współpracy między Uniwersytetem imienia Adama Mickiewicza w Poznaniu i firmą Samsung Research Poland. Ich celem było opracowanie i wdrożenie modeli automatycznej korekty

[3] WRR – *Word Recognition Rate*, zobacz sekcja 3.2.2

błędów i normalizacji wyników rozpoznawania mowy, spełniających wymogi stawiane przez wdrożeniowy kontekst prowadzonych badań, takich jak łatwość interpretacji i modyfikacji działania metody oraz efektywność obliczeniowa umożliwiająca uruchomienie modeli w środowiskach o ograniczonych zasobach obliczeniowych. Istniejące w trakcie prowadzenia badań dotychczasowe rozwiązania problemów korekty i normalizacji nie spełniały tych wymogów, stąd podjęto decyzję o prowadzeniu badań nad rozwiązaniami je spełniającymi.

1.2. Cel i zakres pracy

Cele badawcze postawione przed rozpoczęciem badań, które doprowadziły do powstania niniejszej rozprawy doktorskiej, będące wynikiem ówczesnego stanu wiedzy oraz zapotrzebowania przedsiębiorstwa, były następujące:

- Opracowanie metody korekty błędów systemu rozpoznawania mowy spełniającej wymogi stawiane przez wdrożeniowy kontekst badań (przedstawiony w rozdziale 2), to jest działającej w sposób efektywny i umożliwiającej precyzyjną kontrolę nad jej działaniem (rozdział 4),
- Opracowanie metod normalizacji wyjścia z systemu rozpoznawania mowy, w szczególności przywracania znaków interpunkcyjnych, spełniających wymogi stawiane przez wdrożeniowy kontekst badań (rozdział 2), to jest działających w sposób efektywny i umożliwiający precyzyjną kontrolę nad jej działaniem (rozdział 5).

W rezultacie przeprowadzonych prac badawczych uzyskano następujące wyniki:

- Opracowano metodę korekty błędów "Otaguj i popraw" (opisaną w rozdziale 4.2.1),
- Opracowano metodę normalizacji tekstu "Otaguj i przywróć" (opisaną w rozdziale 5.4),
- Dokonano adaptacji metody "Otaguj i popraw" do analizy wpływu błędów ASR na działanie modeli NLU (opisaną w rozdziale 6.3.1),
- Stworzono zbiór danych służących do rozwoju i ewaluacji modeli korekty błędów ASR (opisany w rozdziale 4.3.3),
- Stworzono zbiór danych służących do rozwoju i ewaluacji modeli NLU odpornych na błędy ASR (6.2.1).

1.3. Metoda badawcza

W badaniach zastosowano metodę eksperymentalną używaną w badaniach nad przetwarzaniem języka naturalnego i nad metodami uczenia maszynowego [74][63][61][9][40][39] polegającą na:

1. pozyskaniu zbioru danych,

2. stworzeniu modeli/metod w oparciu o te dane,
3. ewaluacji ich efektywności przy pomocy wydzielonego zbioru danych testowych.

W metodzie tej istotną rolę odgrywa dobór danych oraz metod ewaluacji. Jeśli autor metody rozwiązującej problem jest również odpowiedzialny za dobór lub wytworzenie danych oraz wybór metod ewaluacji, zachodzi ryzyko popełnienia (być może nieświadomie) błędów metodologicznych polegających na manipulacji danymi i metodami ewaluacji w celu uzyskania lepszych wyników oceny samej metody. Dlatego tak ważną rolę w rozwoju metod i zastosowań uczenia maszynowego stanowią otwarte wyzwania (ang. *Open Challenge* lub *Shared task*) takie jak na przykład Semeval [102] (zadania dotyczące analizy znaczenia języka naturalnego), Poleval [100] (kampania ewaluacyjna narzędzi do przetwarzania języka polskiego) czy zadania w ramach warsztatów WMT [65] (zadania dotyczące tłumaczenia maszynowego). Mając powyższe na uwadze, w ramach prowadzonych badań autor zdecydował się poddać swoje rozwiązania weryfikacji w wyzwaniach Poleval 2020 [100] oraz Poleval 2021 [101], a także był współorganizatorem wyzwań "Open Challenge for Correcting Errors of Speech Recognition Systems" [72] oraz "Center for Artificial Intelligence Challenge on Conversational AI Correctness" [70].

1.4. Znaczenie rozprawy

Problemy automatycznej korekty błędów i normalizacji wyjścia z systemu mowy były wcześniej podejmowane w badaniach naukowych (zob. rozdz. 4.1 i 5.2) i sam pomysł automatyzacji tych procesów nie jest nowatorski. Należy jednak podkreślić wdrożeniowy kontekst badań, który wymagał wypracowania metod spełniających narzucone przez ten kontekst ograniczenia (zob. rozdz. 2). Brak istniejących rozwiązań spełniających te ograniczenia stanowił motywację do podjęcia problemu automatycznej korekty i normalizacji tekstu w kontekście produkcyjnych systemów dialogowych.

Warto również zaznaczyć, że badania były prowadzone w dynamicznie zmieniającym się kontekście naukowym i technologicznym, uwzględniając zawsze aktualny stan badań. Zaproponowane autorskie metody "Otaguj i popraw" oraz "Otaguj i przywróć" wpisywały się w ówczesny rozwój dziedziny, będąc w trakcie powstawania nowatorskimi rozwiązaniami.

Uzyskane wyniki mają również znaczenie z biznesowego punktu widzenia. Zaproponowana metoda "Otaguj i popraw" została wdrożona w systemach dialogowych firmy Samsung, poprawiając jakość oferowanych przez nią usług. Jak pokazują wyniki z rozdziału 4.4.1, zaproponowana metoda umożliwia redukcję ponad 20% błędów popełnianych przez model rozpoznawania mowy na danych przemysłowych, odzwierciedlających warunki, w których pracuje system korekty błędów po jego wdrożeniu. Wyniki te zostały zaprezentowane w publikacji autora [157].

Ta sama metoda korekty błędów została wykorzystana do stworzenia zgłoszenia rozwiązania autora [155] do konkursu Poleval 2020 zad. 1. [100] uzyskując drugie miejsce (opis wyzwania w można znaleźć w rozdziale 4.3.2 a opis wyników w rozdziale 4.4.2). Zaproponowana metoda "Otaguj i przywróć" została wykorzystana przy tworzeniu zgłoszenia [156] w konkursie Poleval 2021 zad. 1 [101] i uzyskała 4. miejsce wykorzystując niepełny zbiór danych trenujących (opis wyników w rozdziale 5.4.3). Metoda "Otaguj i popraw" znalazła również zastosowanie jako część metody służącej do analizy wpływu błędów ASR na działanie modeli NLU (rozdział 6.3.1).

1.5. Struktura pracy

Struktura pozostałej części pracy jest następująca:

Rozdział 2 ukazuje wdrożeniowy kontekst przeprowadzonych badań i wiążące się z nim ograniczenia.

Rozdział 3 przedstawia modele uczenia maszynowego oraz metody ich ewaluacji, wykorzystywane w dalszych rozdziałach pracy.

Rozdział 4 przedstawia zaproponowaną przez autora metodę automatycznej korekty błędów popełnianych przez systemy rozpoznawania mowy "Otaguj i popraw".

Rozdział 5 przedstawia zaproponowaną przez autora metodę automatycznej odwrotnej normalizacji tekstu na potrzeby systemów przetwarzania mowy, w szczególności przywracania interpunkcji, nazwaną "Otaguj i przywróć".

Rozdział 6 opisuje zastosowanie metod opracowanych podczas badań do badania wpływu błędów rozpoznawania mowy na skuteczność modeli rozumienia języka naturalnego (NLU).

Rozdział 7 stanowi podsumowanie całej pracy, omawia płynące z niej wnioski i zarysowuje tematykę przyszłych badań.

Rozdział 2

Wdrożeniowe uwarunkowania badań

Badania opisane w niniejszej pracy były realizowane we współpracy pomiędzy Uniwersytetem imienia Adama Mickiewicza a firmą Samsung Electronics Poland. Były one motywowane zarówno celami naukowymi – rozwojem nauki – jak i biznesowymi – rozwojem produktów oferowanych przez firmę Samsung. Wdrożeniowy charakter prowadzonych prac z jednej strony umożliwił precyzyjne zdefiniowanie problemów badawczych – wynikały one z zapotrzebowania firmy. Z drugiej strony narzucał ograniczenia dotyczące możliwych do zastosowania rozwiązań i stawiał wyzwania z nimi związane. Poniżej zostaną omówione wspomniane ograniczenia i opisana zostanie specyfika prowadzenia badań w środowisku przemysłowym.

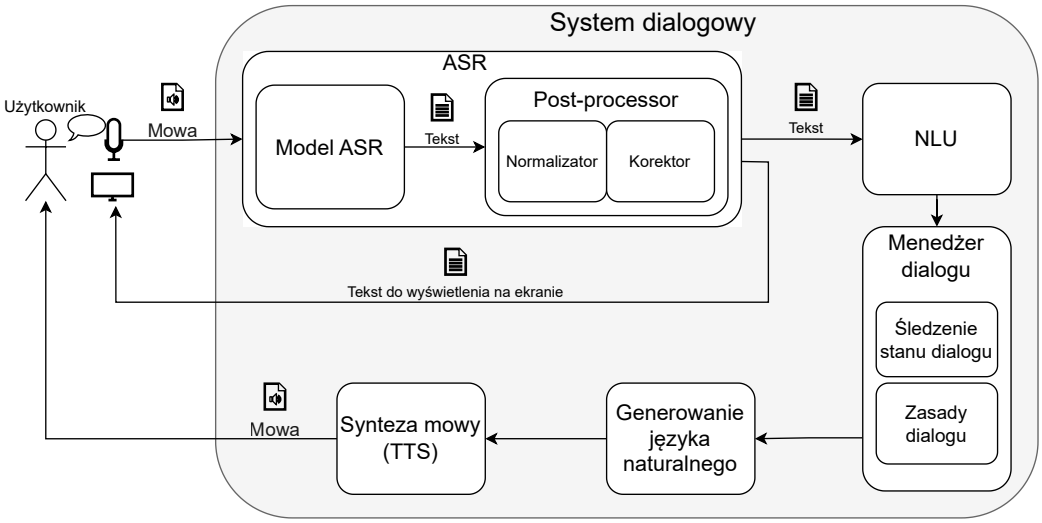
2.1. Specyfika rozwoju agentów dialogowych

Opracowane w ramach przeprowadzonych badań modele korekty, normalizacji i analizy błędów systemu rozpoznawania mowy są częścią większej całości – systemów dialogowych firmy Samsung. System dialogowy (również *agent dialogowy, system konwersacyjny*) [61] umożliwia użytkownikowi prowadzenie rozmowy z komputerem w języku naturalnym. Może ona mieć praktyczny cel (systemy nakierowane na cel), na przykład aktywowanie pewnej funkcji urządzenia, dokonaniu zakupu, odnalezienie informacji, albo może służyć samej rozmowie, na przykład w celach rozrywkowych (tzw. *chatbot*).

System dialogowy standardowo składa się z następujących części [61][148]:

1. Dekoder wejścia – moduł przetwarzający wejście użytkownika i zamieniający je na tekst lub inną reprezentację nadającą się do dalszego przetwarzania. W zależności od modalności systemu wejściem może być tekst pisany na klawiaturze, mowa [27][59], gesty [139], pismo odręczne [27][59], język migowy [51] albo ich kombinacja (systemy multimodalne) [80][27]. W przypadku systemu korzystającego z modalności głosowej (ang. *Spoken dialogue system*) dekoderem wejścia jest moduł rozpoznawania mowy (ASR).
2. Moduł analizy języka naturalnego (ang. Natural Language Understanding (NLU)) – jest odpowiedzialny za interpretację znaczenia wypowiedzi użytkownika w kontekście danego systemu. Znaczenie zostaje zakodowane za pomocą reprezentacji umożliwiającej jej dalsze przetworzenie i wygenerowanie odpowiedzi albo wykonanie przez system akcji. Standardowo [61][144] reprezentacja semantyczna (zwana ramą semantyczną) zawiera

oznaczenie domeny (na przykład ”Music”), zamiaru (ang. *intent*) użytkownika (na przykład. `play_artist`) oraz tak zwane *sloty*, czyli wartości precyzujące zamiar (w postaci klucz:wartość, na przykład: `artist: Queen` albo w postaci odwołania do fragmentów wejściowego zdania).
3. Menedżer dialogu – utrzymuje wewnętrzną reprezentację stanu dialogu, umożliwiając prowadzenie konwersacji składających się z wielu zwrotów wypowiedzi pomiędzy systemem i użytkownikiem i gromadzenie informacji zawartych w tych wypowiedziach w celu realizacji celu użytkownika. Na podstawie utrzymywanego stanu dialogu i przyjętej taktyki prowadzenia dialogu podejmuje decyzje o akcji do podjęcia (na przykład zadanie pytania użytkownikowi albo realizacja polecenia użytkownika).
4. Moduł generowania języka naturalnego (ang. Natural Language Generation (NLG)) – na podstawie informacji zwróconych przez menedżera dialogu generuje odpowiedź skierowaną do użytkownika, sformułowaną w języku naturalnym.
5. Moduł kodowania wyjścia – zamieniający tekst wyjściowy zwrócony przez moduł generowania języka naturalnego na pożądaną modalność – tekst, mowę (poprzez model TTS), język migowy.



Rysunek 2.1: Architektura typowego systemu dialogowego z wyszczególnionym modułem post-processora zawierającym podmoduły normalizacji i korekty błędów

Schemat przedstawiający architekturę systemu dialogowego korzystającego z modalności głosowej, przedstawiono na rysunku 2.1. W przypadku takiego systemu modułem dekodera wejścia jest system rozpoznawania mowy (*ASR*), zaś modułem kodowania wyjścia jest moduł syntezy mowy (*TTS*). Na schemacie przedstawiono wewnętrzną architekturę systemu rozpoznawania mowy, na który składa się model zamiany mowy na tekst i post-processor, realizujący funkcje normalizacji tekstu i korekty błędów.

Użytkownik łączy się z systemem za pomocą urządzenia końcowego (na przykład komputera, urządzenia mobilnego, telewizora, telefonu). Zapewnia ono interfejs do interakcji z systemem, umożliwiając wysyłanie do systemu nagrań mowy lub tekstu oraz wyświetlanie zwróconego przez system tekstu

i odtwarzanie mowy. W zależności od szczegółów implementacyjnych system dialogowy może znajdować się na urządzeniu końcowym [109] (system wbudowany, ang. *embedded, on-device*), albo na dedykowanym serwerze.

Urządzenie końcowe wysyła wypowiedź użytkownika (na przykład "Włącz światło") lub jej fragment, w postaci nagrania dźwiękowego do systemu dialogowego. Nagranie zostaje przekazane do systemu rozpoznawania mowy, gdzie model ASR zamienia je na tekst. Post-processor dokonuje normalizacji i korekty błędów w rozpoznanym tekście i przekazuje wynik w dwa miejsca: na wyświetlacz urządzenia końcowego oraz do modułu NLU. Dzięki otrzymywaniu w trakcie interakcji z urządzeniem wyników rozpoznawania mowy, użytkownik ma kontrolę nad jego działaniem i może przerwać interakcję lub dokonać poprawek, na przykład powtórzyć błędnie rozpoznany wyraz. Wyniki rozpoznawania zwracane przez post-processor do modułu NLU i do użytkownika w razie konieczności mogą się różnić, jeśli moduł NLU oczekuje wejścia znormalizowanego w sposób, który byłby nieprzyjazny dla użytkownika. Na przykład model NLU może być trenowany przy użyciu tekstów niezawierających wielkich liter i przez to może oczekiwać takiego tekstu na wejściu podczas inferencji, dla użytkownika natomiast bardziej czytelna jest forma zawierająca wielkie litery. W takim przypadku post-processor zwróci do urządzenia końcowego hipotezę zawierającą wielkie litery a do modułu NLU przekaże hipotezę, w której wszystkie litery zamieniono na małe.

Model NLU dokonuje klasyfikacji domeny (na przykład `smart_home`), intencji użytkownika (na przykład `turn_light_on`) i wydobywa z tekstu wartości konieczne do realizacji zadania (na przykład: "`location: living_room`"). Moduł menedżera dialogu utrzymuje wewnętrzną reprezentację stanu dialogu za pomocą dotychczasowych wypowiedzi użytkownika i wartości zwróconych przez model NLU. Na podstawie stanu dialogu i taktyki prowadzenia dialogu (ang. *dialog policy*) menedżer podejmuje decyzję o akcji do podjęcia. Może być to na przykład zadanie użytkownikowi pytania, które pozwoli doprecyzować jego intencję (na przykład: "`które światło chcesz włączyć?`") albo wykonanie polecenia (na przykład włączenie światła).

Systemy dialogowe mogą wykorzystywać różne modalności na swoim wejściu i wyjściu – od najprostszych agentów porozumiewających się za pomocą tekstu po złożone systemy rozmawiające z użytkownikiem za pomocą głosu. Te ostatnie stały się powszechne za sprawą takich agentów jak Apple Siri, Google Assistant, Amazon Alexa czy Bixby firmy Samsung. Stosowanie modalności głosowej do komunikacji z użytkownikiem zapewnia bardziej naturalną, zbliżoną do ludzkiej interakcję z systemem, jednak wymaga budowy złożonego systemu, zawierającego w sobie komponenty rozpoznawania oraz syntezy mowy. Używanie modalności głosowej może wpływać pozytywnie na sprawność realizacji założonego celu użytkownika [103], ale złożoność systemu oznacza, że każdy jego komponent może wprowadzić błąd prowadzący do ostatecznego niepowodzenia w realizacji celu. Na przykład błędy wprowadzone na etapie rozpoznawania mowy mogą mieć wpływ na skuteczność modelu NLU [120] (rozwinięcie tego problemu zostało przedstawione w rozdziale 6).

Rozwój systemu rozpoznawania mowy, w tym jego części takich jak post--processor, musi uwzględniać kontekst systemów, z którymi on współpracuje. Wymaga to między innymi:

- możliwości precyzyjnej kontroli sposobu działania rozwijanych modeli tak, aby móc dostosować zwracane wyniki do wymagań modułów korzystających ze zwracanych przezeń rezultatów,
- automatyzacji procesów trenowania modeli, tak, żeby mogły być one na bieżąco dostosowywane do zmieniających się danych trenujących pochodzących z innych modeli (na przykład danymi trenującymi dla post-processora mogą być dane zwracane przez moduł zamiany mowy na tekst),
- zautomatyzowanej kontroli nad jakością wyników dostarczanych przez modele, żeby mogły być one na bieżąco ewaluowane przy użyciu zmieniających się danych testowych pochodzących z innych modeli (na przykład danymi testowymi dla post-processora mogą być dane zwracane przez moduł zamiany mowy na tekst).

Prowadzone prace badawcze koncentrowały się na spełnieniu tych wymogów względem modeli korekty błędów i normalizacji, przy jednoczesnym zapewnieniu ich wysokiej jakości oraz weryfikowalności wyników.

2.2. Specyfika pracy w środowisku produkcyjnym

Podczas rozwoju każdego z komponentów systemu dialogowego należy brać pod uwagę kontekst całego systemu – komponenty są od siebie wzajemnie zależne i zmiana w jednym może rodzić nieprzewidziane komplikacje w działaniu pozostałych części [119]. Rozwój systemu w środowisku produkcyjnym oznacza ciągłe zmiany w powiązanych ze sobą komponentach. Na przykład zmiany w danych użytych do trenowania modelu rozpoznawania mowy wpływają na zwracany przez ten model tekst. Pewne błędy przestają się pojawiać w rozpoznawanym tekście, za to pojawiają się nowe. Wpływa to na wydajność modułów znajdujących się w potoku przetwarzania za modelem rozpoznawania mowy, w tym na wydajność post-processora. Aby modele będące częścią post-processora pracowały dobrze z nowym modelem ASR, muszą być wytrenowane od nowa, co potencjalnie może doprowadzić do pojawienia się nowych błędów wprowadzonych przez te modele i propagowania ich dalej w potoku przetwarzania, na przykład do modułu NLU. Ręczna kontrola jakości tak złożonych systemów jest bardzo wymagająca i czasochłonna, dlatego powinna ona podlegać automatyzacji. Poniżej zostanie przedstawiony cykl rozwoju modeli post-processora w kontekście rozwoju systemu rozpoznawania mowy będącego częścią systemu dialogowego.

2.2.1. Cykl rozwoju modeli post-processora

Post-processor systemu ASR, to moduł działający na wyjściu modelu rozpoznawania mowy. Otrzymuje surowy tekst zwracany przez moduł zamiany mowy na tekst i ma za zadanie nadać mu odpowiednią formę (normalizacja tekstu), a także naprawić błędy wprowadzone przez model ASR. Dane trenujące modele post-processora powinny zatem zawierać hipotezy systemu ASR oraz oczekiwane wyjście, niezawierające błędów i poddane normalizacji. Źródłem takich danych mogą być wyniki ewaluacji samego modelu zamiany mowy na tekst. Zbieranie danych ewaluacyjnych modelu ASR w środowisku przemysłowym przebiega zazwyczaj na jeden z dwóch sposobów:

- wykorzystuje istniejący korpus mowy, zawierający nagrania wypowiedzi oraz ich prawidłowe transkrypcje – nagranie jest podawane na wejście modelu ASR a zwrócona przezeń hipoteza jest porównywana z prawidłową transkrypcją znajdującą się w korpusie;
- może wykorzystywać dane rejestrowane podczas działania systemu w środowisku produkcyjnym, czyli nagrania wypowiedzi użytkowników oraz zwracane dla nich hipotezy, które dopiero post-factum podlegają anotacji – nagrania są odsłuchiwane przez człowieka dokonującego transkrypcji wypowiedzi.

W wyniku wykorzystania obu powyżej wymienionych źródeł otrzymuje się dla każdej wypowiedzi poniższe dane:

- sygnał dźwiękowy zawierający wypowiedź skierowaną do systemu ASR,
- hipotezę – transkrypcję dokonaną przez moduł ASR,
- transkrypcję referencyjną – prawidłowy zapis wypowiedzi. Może on występować w kilku formach różniących się od siebie sposobem normalizacji,
- dodatkowe metadane.

Powyższe dane, uzupełnione o wartości metryk wyliczonych w wyniku porównania hipotez i oczekiwanych transkrypcji są zapisywane w bazie danych, co umożliwia ich późniejszą analizę i porównywanie między sobą rozwijanych modeli rozpoznawania mowy.

Spośród opisanych powyżej wyników ewaluacji systemu ASR, dane składające się z par hipoteza-transkrypcja referencyjna są przydatne do rozwoju i ewaluacji modeli normalizacji i korekty błędów realizowanych w post-processorze (hipoteza ASR stanowi jego wejście a referencyjna transkrypcja oczekiwane wyjście).

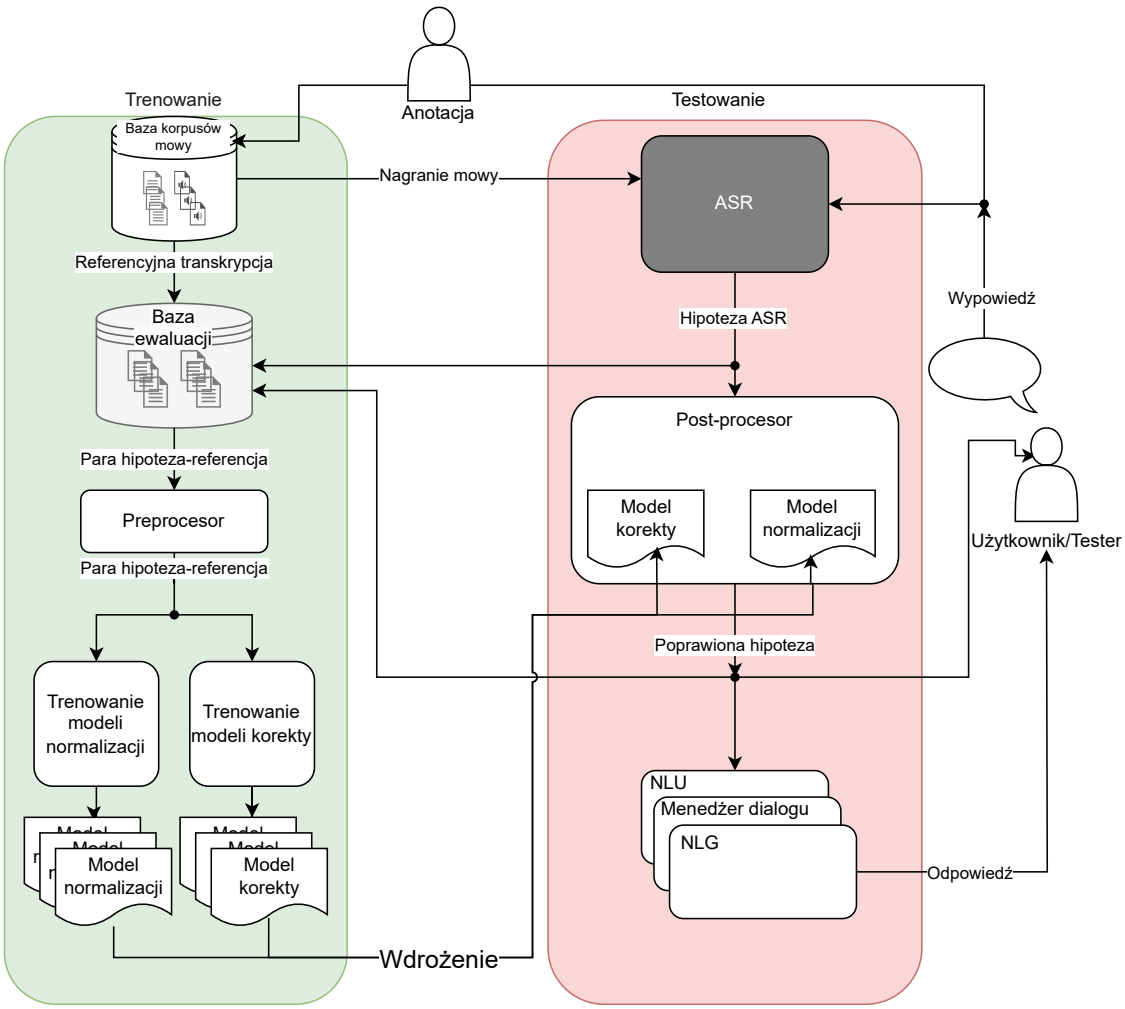
Cykl rozwoju modeli post-processora w ramach systemu dialogowego działającego w środowisku produkcyjnym pokazano na schemacie 2.2. Elementy widoczne na schemacie to:

Baza korpusów mowy – służy do gromadzenia i udostępniania korpusów mowy na potrzeby trenowania i ewaluacji modeli ASR;

Baza ewaluacji – przechowuje wyniki działania modeli rozpoznawania mowy i post-processora wraz z odpowiadającymi im oczekiwanymi wynikami (referencyjnymi transkrypcjami);
Preprocesor – przetwarza tekst hipotez i transkrypcji referencyjnych dostosowując go do wymogów stawianych przez zadanie trenowania;
Trenowanie modeli normalizacji/korekty – na podstawie danych wejściowych, przy zastosowaniu zadanych parametrów, tworzy statystyczny model normalizacji/korekty;
Model normalizacji – model statystyczny służący do przeprowadzania odwrotnej normalizacji tekstu, na przykład przywracania w nim znaków interpunkcyjnych albo wielkich liter;
Model korekty – model statystyczny służący do poprawy błędów w wejściowym tekście;
ASR – model rozpoznawania mowy wraz ze środowiskiem uruchomieniowym, zamieniający dźwiękowy sygnał mowy na odpowiadający mu zapis tekstowy – transkrypcję, zwaną też hipotezą modelu ASR;
Post-processor – środowisko uruchomieniowe modeli normalizacji i korekty błędów. Korzystając z tych modeli dokonuje korekty błędów i normalizacji w tekście zwracanym przez model ASR;
NLU, Menedżer dialogu, NLG – moduły systemu dialogowego przetwarzające zwracaną przez post-processor hipotezę;
Anotacja – anotacja nagrań mowy, w wyniku której powstaje referencyjna transkrypcja wypowiedzi.

Po lewej stronie schematu pokazano proces trenowania modeli, po prawej ich ewaluacji. Elementy systemu ukazane po prawej stronie są udostępnione w sposób umożliwiający dostęp do agenta dialogowego dla użytkownika końcowego lub osób zajmujących się testowaniem systemu (testerów). Elementy ukazane po lewej stronie są dostępne jedynie wewnątrz organizacji rozwijającej system. Udostępnienie modeli, poprzez przekazanie ich z lewej części do prawej nazywa się wdrożeniem. Proces wdrożenia obejmuje zazwyczaj wielostopniowe testowanie i ewaluację elementów mających podlegać wdrożeniu. Elementy te, takie jak modele uczenia maszynowego, zestawy reguł czy programy i biblioteki komputerowe są w trakcie wdrożenia kopiowane i uruchamiane w środowisku, w którym są potem dostępne jako część uruchomionego systemu. W przypadku systemów dialogowych Środowiskiem tym może być serwer (systemy działające w chmurze) albo urządzenie końcowe, na przykład urządzenie mobilne (systemy działające na urządzeniu, ang. *on-device*). Ze względu na konieczność zachowania czytelności, schemat jest uproszczony względem rzeczywistego procesu. W rzeczywistości wdrożenie jest przeprowadzane etapami – nowe modele są początkowo wdrażane w środowisku wewnętrznym, do którego dostęp mają tylko inżynierowie rozwijający system. W wyniku przeprowadzanych ewaluacji dokonuje się selekcji modeli osiągających najlepsze wyniki i te modele są wdrażane w środowiskach dostępnych coraz szerszej grupie osób, od osób odpowiedzialnych za testowanie systemu aż po instancje systemu udostępnione finalnym użytkownikom (środowisko produkcyjne).

Na dane używane do ewaluacji i trenowania modeli ASR składają się nagrania



Rysunek 2.2: Przepływ danych w procesie rozwoju modeli korekty błędów i normalizacji ASR

mowy wraz z odpowiadającymi im referencyjnymi transkrypcjami. Zbiór takich par nagranie-transkrypcja nazywamy *korpusem mowy*. Podział nagrań na korpusy może być dokonany ze względu na różne cechy, na przykład pochodzenie, warunki nagrywania, tematykę wypowiedzi. Zbiór korpusów to *baza korpusów mowy*. Korpusy w bazie podlegają rozszerzaniu i zmianom. Podczas użytkowania systemu przez użytkowników i przeprowadzania wewnętrznych testów wybrane nagrania wypowiedzi skierowanych do systemu są zapisywane w bazie wraz z transkrypcją wprowadzoną przez *anotatorów*.

Zbieranie danych ewaluacyjnych modelu ASR polega na rozpoznaniu nagrań z wybranych korpusów mowy przez ewaluowany system ASR i zapisaniu wynikowych hipotez, wraz z odpowiadającymi im referencyjnymi transkrypcjami, w bazie ewaluacji w celu ich porównania. Oprócz hipotez pochodzących z modelu ASR w bazie są zapisywane również hipotezy przetworzone przez moduł *post-processora*. Ewaluacja systemu rozpoznawania mowy może być wywołana ręcznie albo automatycznie, po pojawieniu się nowej wersji testowa-

nych modułów. Analiza danych zawartych w tej bazie umożliwia identyfikację "słabych punktów" modeli ASR i post-processora i podjęcie decyzji dotyczących kierunku rozwoju poszczególnych modułów systemu, co zostało opisane w pracy, której współautorem był autor rozprawy [57].

Dane zapisane w *bazie ewaluacji* są używane do trenowania modeli post-processora, takich jak *modele korekty* błędów czy *normalizacji*. Trenowanie może być uruchamiane manualnie albo automatycznie, po pojawieniu się nowych wyników ewaluacji modelu ASR. Zbiór par referencyjna transkrypcja – hipoteza ASR (potencjalnie zawierająca błędy), po wstępnym przetworzeniu w *preprocesorze*, stanowi dane wejściowe dla procesów trenowania tych modeli. Poprzez dobór parametrów trenowania oraz danych trenujących dąży się do uzyskania modeli post-processora dających jak najlepsze wyniki podczas wewnętrznej ewaluacji (bez użycia pozostałych komponentów systemu dialogowego). Wyniki zwracane przez ten model są porównywane z wynikami zwracanymi przez ostatni wdrożony model w celu detekcji przypadków, dla których nastąpił spadek skuteczności modelu. Analiza takich błędów regresji ma na celu identyfikację ich źródeł i eliminację, na przykład poprzez poprawę błędnych danych trenujących, zmianę parametrów albo modyfikację rozkładu przypadków z różnych korpusów w danych trenujących. Kiedy rozwijany model spełnia określone wymagania, na przykład daje lepsze uśrednione wyniki od poprzednio wdrożonego modelu, jednocześnie nie powodując pogorszenia pojedynczych przypadków testowych, podjęta pozostaje decyzja o jego wdrożeniu. Początkowo wdrożenie następuje w środowisku wewnętrznym. Model staje się wtedy częścią testowej instancji systemu dialogowego i podlega ewaluacji razem z jego pozostałymi komponentami, której wyniki są zapisywane do *bazy ewaluacji*. Dzięki temu możliwe jest porównywanie różnych wersji modelu i wybór jednej z nich do wdrożenia w środowiskach produkcyjnych.

2.2.2. Ograniczenia pozafunkcjonalne

Modele uczenia maszynowego przeznaczone do wdrożenia w środowisku produkcyjnym podlegają ograniczeniom pozafukcjonalnym. Ograniczenia te dotyczą zasobów używanych przez modele oraz przez środowisko uruchomieniowe służący do ich uruchamiania, takich jak:

- zajmowana przez modele przestrzeń dyskowa,
- zajmowana przez modele pamięć operacyjna,
- czas przetworzenia przez model jednego przypadku testowego, na przykład zdania,
- czas potrzebny na inicjalizację modelu, czyli załadowanie go do pamięci.

W przypadku systemów działających "w chmurze", czyli na serwerach, z którymi urządzenie końcowe użytkownika komunikuje się za pomocą sieci internet, konieczne jest umożliwienie równoległego przetwarzania zapytań pochodzących od wielu użytkowników. Serwery posiadają relatywnie duże zasoby obliczeniowe [135], ale muszą być one dzielone między wszystkich użytkowników obsługiwanych przez dany serwer.

Alternatywą do uruchamiania systemu dialogowego w chmurze jest uruchomienie go bezpośrednio na urządzeniu użytkownika. Takie rozwiązanie ma wiele zalet, między innymi [47][18][135]:

- Zapewnia większą prywatność – dane użytkownika nie muszą opuszczać urządzenia.
- Nie wymaga utrzymywania serwerów.
- Umożliwia działanie agenta bez dostępu do internetu.
- jest bardziej responsywne, dzięki eliminacji opóźnienia związanego z transferem danych przez internet między serwerem a urządzeniem końcowym.
- Nie wymaga kompresji sygnału mowy przed przesłaniem go na serwer, która może wpływać negatywnie na wyniki rozpoznawania.

Takie podejście wiąże się jednak z dodatkowymi ograniczeniami [135][47][18]:

- Urządzenia końcowe posiadają zazwyczaj mniejsze zasoby obliczeniowe w porównaniu z serwerami co ogranicza użycie modeli wymagających zasobów obliczeniowych przekraczających możliwości urządzenia.
- Wykonywanie wymagających obliczeniowo operacji na urządzeniu mobilnym powoduje zwiększenie użycia energii i co za tym idzie skrócenie czasu pracy urządzenia na baterii, dlatego dąży się do optymalizacji modeli wykonywanych na urządzeniu końcowym pod tym kątem.
- Ze względu na różnorodność urządzeń końcowych konieczne jest zapewnienie działania systemu w środowiskach różniących się od siebie zasobami obliczeniowymi oraz wersją zainstalowanego oprogramowania.
- W przeciwieństwie do systemów wdrażanych na serwerach, brak pełnej kontroli nad wdrożoną wersją systemu – w przypadku wdrożenia wersji systemu zawierającej błędy nie da się w prosty sposób zastąpić jej nową wersją u wszystkich użytkowników, co jest możliwe w przypadku systemu wdrożonego na serwerze.

Rozwijane w tej pracy modele były przygotowane z myślą o wdrożeniu zarówno w chmurze jak i na urządzeniach końcowych. Dlatego przy ich tworzeniu ważnym kryterium była efektywność działania wyrażona w czasie potrzebnym na przetworzenie jednego zdania (wyniki przedstawiono w rozdziale 4.4.1) oraz zajętość pamięci.

Rozdział 3

Modele uczenia maszynowego i metody ewaluacji

W niniejszym rozdziale przedstawione zostały użyte podczas prowadzonych badań modele uczenia maszynowego, w szczególności modele tagowania sekwencji, a także metody ewaluacji modeli korekty błędów i tagowania.

3.1. Modele uczenia maszynowego użyte w pracy

Problem tagowania sekwencji jest jednym z klasycznych zadań przetwarzania języka naturalnego. Model tagowania częściami mowy (POS-tagger) jest jedną ze standardowych części klasycznego potoku przetwarzania w NLP (przegląd metod można znaleźć w [15]). Wraz z pojawieniem się głębokich sieci neuronowych i wzrostem ilości danych treningowych ich znaczenie spadło, ponieważ większość zadań jest realizowana w paradygmacie end-to-end. W paradygmacie tym zakłada się, że potrzebne do rozwiązania docelowego problemu (na przykład tłumaczenia maszynowego albo rozumienia języka (NLU)) reprezentacje, są konstruowane w warstwach ukrytych sieci przez algorytm uczący sieć i nie ma potrzeby stosowania osobnych modeli. Niemniej modele tagowania wciąż są rozwijane – zadanie tagowania może stanowić docelowe zadanie albo może być wykorzystane jako krok pośredni umożliwiający lepszy wgląd i kontrolę nad rozwijanym systemem, jak ma to miejsce w przypadku omawianych modeli korekcji błędów.

Problem tagowania jest rodzajem bardziej ogólnego problemu przepisywania jednej sekwencji na drugą, z dodatkowym ograniczeniem, że obie sekwencje muszą być tej samej długości. W związku z tym zastosowanie znajduje w nim wiele metod sequence-to-sequence stosowanych w takich problemach jak tłumaczenie maszynowe, sumaryzacja, odpowiadanie na pytania itp.

Poniżej zostaną opisane metody tagowania wykorzystywane w niniejszej pracy. Opis rozpoczyna się od prostych metod indukcji reguł a kończy na złożonych metodach wykorzystujących pretrenowane modele typu Transformer [140].

Tager Brilla

Tager Brilla [11] to klasyczny tager regułowy, oryginalnie wykorzystany jako POS-tagger. Reguły tagowania są automatycznie indukowane na podstawie danych trenujących. W kolejnych krokach tager inkrementacyjnie polepsza

swoje działanie, ulepszając listę reguł tagowania na podstawie popełnianych przez siebie błędów. Korpus należy podzielić na 3 podzbiory:

1. Zbiór trenujący (oryginalnie 90% korpusu),
2. Zbiór służący do indukcji reguł poprawek (5% korpusu),
3. Zbiór testowy (5% korpusu).

W pierwszym kroku każdemu wyrazowi w zbiorze trenującym jest przypisany najbardziej prawdopodobny tag na podstawie częstości przypisania tagów do tego wyrazu zbiorze trenującym, bez uwzględnienia kontekstu. Na przykład, gdyby tager brilla wytrenować na danych z korpusu NKJP [113] we wszystkich poniższych zdaniach wyraz "rad" zostanie otagowany jako rzeczownik, ponieważ w korpusie najczęściej występuje jako rzeczownik:

Pierre Curie wykryła dwa pierwiastki promieniotwórcze: polon i rad.

A Pan, ile skorzystał z rad swoich pedagogów?

Rad jestem, że skrzyżowały się nasze drogi.

W następnym kroku algorytm porównuje przypisane w pierwszym kroku tagi z poprawnymi tagami z podzbioru indukcji reguł poprawek. Każdy błąd polegający na zastosowanie tagu a w miejsce tagu b jest zliczany w postaci trójek $< tag_a, tag_b, N >$, gdzie N oznacza częstość danej pomyłki. Po zliczeniu wszystkich błędów algorytm, zaczynając od najbardziej licznych, generuje reguły zmieniające domyślne tagowanie. Reguły są tworzone na podstawie szablonu:

"Zamień tag A na tag B jeśli *warunek*"

gdzie *warunek* może być postaci:

1. Poprzedzające/następne słowo jest otagowane tagiem z,
2. Słowo dwa słowa do przodu/tyłu jest otagowane tagiem z,
3. Jedno z dwóch poprzedzających/następnych słów jest otagowane tagiem z,
4. Jedno z trzech poprzedzających/następnych słów jest otagowane tagiem z,
5. Poprzedzające słowo jest otagowane takiem z a następne słowo tagiem w,
6. Poprzedzające/następne słowo jest otagowane tagiem z a słowo dwa słowa wstecz/do przodu jest takiem w,
7. Bieżące słowo jest/nie jest zapisane wielką literą,
8. Poprzedzające słowo jest/nie jest zapisane wielką literę.

Szablony i warunki służące do tworzenia reguł mogą być dostosowywane do konkretnego problemu, powyżej podano oryginalne użyte w publikacji Brilla. Na przykład, przy zastosowaniu innego zestawu warunków, dla przytoczonych powyżej 3 przykładowych zdań algorytm mógłby wyprodukować regułę:

Zamień tag "NOUN" na "ADJ", jeśli następne słowo zaczyna się od "jest"

Dla każdej trójki $< tag_a, tag_b, N >$ obliczany jest zysk netto z zastosowania reguły powstałej przez użycie jednego z powyższych warunków. Zysk netto jest liczony jako liczba poprawionych tagów minus liczba tagów, które zostały błędnie otagowane przez zastosowanie danej reguły. Reguła z najwyższym zyskiem netto zostaje dodana do listy reguł-poprawek i zastosowana na zbiorze. Procedura jest kontynuowana dla następnych trójek, aż osiągnięta zostanie zakładana liczba reguł.

Podczas inferencji tagowanie polega na przypisaniu każdemu wyrazowi w pierwszym kroku najbardziej prawdopodobnego tagu na podstawie zbioru trenującego a następnie zastosowaniu listy reguł poprawiających tagowanie.

Tager Brilla już w momencie jego stworzenia stanowił próbę demonstracji, że proste, regułowe podejście jest w stanie osiągnąć dobre wyniki i że metody stochastyczne nie są jedynym rozwiązaniem problemu tagowania. Jest on bardzo łatwy w implementacji, jest wydajny obliczeniowo i jego działanie łatwo zinterpretować i dostosować. W zaprezentowanych eksperymentach jest stosowany jako rozwiązanie bazowe, z założeniem, że bardziej wyrafinowane metody powinny osiągnąć znacznie lepsze wyniki.

CRF

Conditional Random Field (Warunkowe Pola Losowe) [73] to rodzina dyskryminatywnych modeli probabilistycznych służących do etykietowania sekwencji. Mogą być postrzegane jako uogólnienie generatywnych modeli HMM (Hidden Markov Models). W odróżnieniu od nich, modelują prawdopodobieństwo warunkowe sekwencji przewidywanych stanów (na przykład etykiet) warunkowane sekwencją obserwacji (na przykład słów), a nie ich wspólne prawdopodobieństwo.

Formalna definicja liniowego wariantu CRF (ang. *linear-chain CRF* przedstawia się następująco:

Niech:

- Y będzie losowym wektorem wartości wyjściowych (np. etykiet części mowy)
- X będzie losowym wektorem wartości wejściowych (obserwacji, np. wyrazów)
- $\mathcal{R} = \{f_k(y, y', \mathbf{x}_t)\}_{k=1}^{K}$ będzie zbiorem funkcji cech przyjmujących wartości rzeczywiste
- $\theta = \{\theta_k\} \in \Re^K$ będzie wektorem parametrów (wag funkcji cech)
- T to długość wektora wartości wejściowych/wyjściowych

Wówczas liniowe warunkowe pole losowe (ang. *linear chain conditional random field* jest rozkładem $p(\mathbf{y} \mid \mathbf{x})$ w postaci:

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \exp\left\{ \sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, \mathbf{x}_t) \right\}, \tag{3.1}$$

gdzie $Z(\mathbf{x})$ jest funkcją normalizującą:

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \prod_{t=1}^{T} \exp \left\{ \sum_{k=1}^{K} \theta_k f_k \left(y_t, y_{t-1}, \mathbf{x}_t \right) \right\}.$$

Stworzenie modelu CRF wymaga od człowieka doboru lub konstrukcji funkcji cech f_k. Trenowanie modelu polega na optymalizacji wektora parametrów θ tak, żeby maksymalizować prawdopodobieństwo zwracane przez wzór 3.1 dla danych trenujących.

CRF są wykorzystywane do tagowania sekwencji, ponieważ potrafią dobrze modelować zależności między elementami w sekwencji. Zanim metody neuronowe stały się dominującymi metodami modelowaina sekwencj, warunkowe pola losowe były powszechnie używane w zadaniach takich jak POS-tagging i NER (Named Entity Recognition - Rozpoznawanie jednostek nazwanych). Obecnie CRF są stosowane jako ostatnia warstwa operująca na wyjściu sieci neuronowej (sieć neuronowa spełnia rolę funkcji cech)[3].

Trudnością związaną z użyciem CRF jest właściwa konstrukcja i dobór funkcji cech. Cechy wybrane do zadania tagowania operacjami edycyjnymi zostały przedstawione w sekcji 4.4.

BERT

BERT (ang. *Bidirectional Encoder Representations from Transformers*) [28] to neuronowy model języka, wykorzystujący bloki typu Transformer [140]. Jest on pretrenowanym na dużych zbiorach czystych danych tekstowych przy pomocy dwóch zadań niewymagających anotacji danych. Może być potem dotrenowany (ang. *finetuned*) na potrzeby konkretnych zadań NLP.

Pierwsze zadanie użyte do pretrenowania modelu, MLM (ang. *Masked LM*), polega na zamaskowaniu 15% losowych tokenów w wejściu spejcalnym tokenem $[MASK]$ i zgadywaniu ich przez sieć. Zadanie to, w przeciwieństwie do powszechnie używanego poprzednio zadania przewidywania następnego (bądź poprzedniego) wyrazu, skłania sieć do wytworzenia reprezentacji dwukierunkowych zależności między wyrazami. Drugie zadanie, za pomocą którego sieć jest wstępnie trenowana, to NSP (ang. *Next Sentence Prediction*. Sieć otrzymuje na wejściu dwa fragmenty tekstu oddzielone specjalnym tokenem separatora $[SEP]$. Połowa z tych par to fragmenty występujące po sobie w korpusie a połowa to losowo wybrane fragmenty z różnych części korpusu. Zadaniem sieci jest klasyfikacja binarna odpowiadająca na pytanie, z którą z tych dwu sytuacji mamy do czynienia.

Oba powyższe zadania mogą być wykonane na bardzo dużych zbiorach danych, ponieważ wymagają jedynie samego tekstu, bez konieczności jego anotacji. Dzięki dużym ilościom danych i głębokiej architekturze sieci (24 warstwy), podczas pretrenowania na opisanych powyżej zadaniach w warstwach ukrytych

sieci wykształca się reprezentacja syntaktycznych i semantycznych właściwości wyrazów i zdań [117]. Umożliwia to dotrenowanie takiej pretrenowanej sieci na konkretnym zadaniu NLP, poprzez dodanie jednej wyjściowej warstwy typu feed-forward i dostosowywaniu parametrów sieci za pomocą relatywnie małego zbioru danych dla konkretnego zadania. Dotrenowanie, w przeciwieństwie do pretrenowania, zajmuje relatywnie mało czasu (ze względu na mniejsze rozmiary zbioru trenującego i szybszą zbieżność trenowania na pretrenowanej sieci). Twórcy BERT dotrenowali sieć na 11 różnorodnych zadaniach NLP, na większości z nich uzyskując nowe wyniki *State of the art* a na pozostałych osiągając wyniki zbliżone do wyników SOTA (State of the art) [28].

Tokenizacja na wejściu modelu jest przeprowadzona za pomocą metody Wordpiece [149] (bardzo zbliżonej do wcześniej zaproponowanej metody BytePair encoding [122]) - dzieli ona wyrazy na fragmenty w taki sposób, żeby przy zadanej wielkości słownika fragmentów zminimalizować liczbę tokenów użytych do podziału danych w korpusie trenującym. Algorytm rozpoczyna od podzielenia wyrazów na jednoznakowe fragmenty i następnie łączy je ze sobą w pary tworząc większe fragmenty aż do osiągnięcia zadanej wielkości słownika. Dzięki temu podejściu jest możliwe reprezentowanie wszystkich potencjalnych słów za pomocą skończonego słownika fragmentów, jednocześnie zmniejszając długość wynikowych sekwencji tokenów.

Po przeprowadzeniu tokenizacji, tokeny są wektoryzowane za pomocą wyuczonych reprezentacji wektorowych (ang. *embeddings*). Do wektorów tych są dodawane dwa dodatkowe wektory: pozycji (ang. *position embedding*) oraz segmentacji (ang. *segmentation embedding*). Pierwszy z nich, wektor pozycji, zawiera informację o pozycji tokena w zdaniu. Informacja ta jest niedostępna w inny sposób ze względu na nierekurencyjną naturę bloków typu transformer i musi być dodana, żeby zachować informację o kolejności wyrazów w wejściowym ciągu [37][140]. Wektor segmentacji zawiera natomiast informację o podziale wejścia na części, przydatną w zadaniu NSP.

Oryginalny model BERT był wytrenowany na tekstach w języku angielskim, później pojawiły się jego wielojęzyczne wersje, a także wersje dla różnych języków, między innymi francuskiego [90], niemieckiego [13], hiszpańskiego [12] i polskiego [96].

Flair

Flair [1] to biblioteka udostępniająca narzędzia do trenowania i wykorzystywania modeli dla wielu problemów NLP, w tym tagowania sekwencji. Umożliwia ona łatwe łączenie istniejących modeli w bardziej złożone potoki przetwarzania. Biblioteka ta osiąga szczególnie dobre wyniki na zadaniach tagowania takich jak NER i POS-tagging. Poza możliwością łączenia istniejących modeli i dotrenowania sieci, bibltioteka udostępnia również unikalne dla niej kontekstowe reprezentacje słów (ang. *Contextual Word Embeddings* [3]). Powstają one poprzez zapisanie wartości z warstwy ukrytej rekurencyjnej sieci neuronowej typu LSTM [50], zawierającej model języka wytrenowany na

znakach z dużego korpusu tekstowego. Dzięki temu otrzymane reprezentacje wektorowe uwzględniają kontekst wyrazu, co sprawia, że ten sam wyraz będzie miał inną reprezentację wektorową w różnych kontekstach. Ułatwia to między innymi odróżnienie homonimów.

XLM-RoBERTa

W eksperymentach opisanych w rozdziale 6 użyty został model analizy języka naturalnego (ang. *NLU - Natural Language Understanding*) wytrenowany w oparciu o duży, pretrenowany model językowy XLM-RoBERTa [21]. Jest to, podobnie do opisywanego powyżej modelu BERT [28], neuronowy model języka korzystający z architektury Transformer, wytrenowany przy użyciu zadania MLM (Masked Language Model), polegającego na zgadywaniu przez sieć zamaskowanych wyrazów w surowym, nieanotowanym tekście. Cechą wyróżniającą model XLM-RoBERTa jest użycie do jego trenowania bardzo dużego zbioru danych, zawierającego teksty w 100 językach. Dzięki temu model ten jest wielojęzyczny, co sprawia, że osiąga dobre wyniki nawet na językach, które miały relatywnie skromną reprezentację w zbiorze trenującym. Wyniki modelu w wielu zadaniach NLP, na których był ewaluaowany są lepsze od wcześniej prezentowanych modeli wielojęzycznych takich jak mBERT [28] czy XLM [23].

3.2. Metody ewaluacji

Dobór odpowiedniej metody oceny modeli uczenia maszynowego, stosowanej podczas ich rozwoju, jest niezwykle istotny dla ich praktycznej użyteczności. Jeśli użyte podczas prac nad modelami metryki nie odzwierciedlają dobrze biznesowych uwarunkowań, w których będą one stosowane, możemy poświęcić wiele czasu i zasobów na ich optymalizację w niewłaściwym kierunku. Poniższa sekcja omawia stosowane w badaniach metody ewaluacji modeli korekty błędów rozpoznawania mowy oraz przedstawia zakres stosowania każdej z nich.

3.2.1. Normalizacja na potrzeby ewaluacji

Ewaluację większości zadań przetwarzania języka naturalnego zwracających na wyjściu tekst (jak np. tłumaczenie maszynowe, rozpoznawanie tekstu pisanego (OCR) czy rozpoznawanie mowy) przprzeprowadza się porównując hipotezę zwróconą przez system ze zdaniem referencyjnym, na przykład transkrypcją pochodzącą z korpusu mowy. Przed obliczeniem wartości metryki oba porównywane ciągi znaków należy znormalizować, żeby uniknąć wpływu nieistotnych, w kontekście danej ewaluacji, różnic między ciągami na wartość metryki.

W zależności od użytych danych treningowych model ASR może zwracać na wyjściu:

- tekst znormalizowany – zawierający jedynie pełne formy słowne wyrażeń, zapisane tak, jak zostały wypowiedziane. Taki tekst nie zawiera żadnych słów niestandardowych[1], na przykład:
  ustaw przypomnienie kupić dwa kilogramy truskawek na czternastą trzydzieści
- tekst odwrotnie znormalizowany – pełna docelowa forma pisana, na przykład:
  Ustaw przypomnienie: "Kupić 2kg. truskawek" na 14:30
- tekst mieszany – zawierający obie formy, na przykład:
  ustaw przypomnienie kupić 2 kilogramy truskawek na czternastą 30

Standardowo dąży się do tego, żeby system ASR zwracał wyniki w pierwszej z powyższych form, jednak w praktyce zdarza się, że w wyniku występowania w danych trenujących form odwrotnie znormalizowanych, w niektórych przypadkach zwraca tekst mieszany. Istnieją też modele trenowane na tekście odwrotnie znormalizowanym, które od razu zwracają poprawnie znormalizowaną formę pisaną, na przykład trenowany na surowych (niepoddanych obróbce) danych z internetu model Whisper [114]. Użyty do ewaluacji systemu ASR korpus mowy może posiadać zdania referencyjne w jednej z trzech wymienionych powyżej form. W zależności od tego, jaką formę zwraca model ASR, jaka forma jest obecna w zdaniach referencyjnych w korpusie oraz wymagań ewentualnych dalszych modeli korzystających z wyników działania systemu ASR (na przykład modeli Natural Language Understanding (NLU)), normalizacji mogą podlegać:

- wielkość liter – wszystkie znaki w porównywanych ciągach można zamieniać na ich małe lub wielkie odpowiedniki. Wybór między zamianą znaków na wielkie lub małe litery jest arbitralny, ale musi być konsekwentnie stosowany. Zamiana liter na wielkie ma tę zaletę, że w jej przypadku łatwo zauważyć, że tekst został znormalizowany, nie zawiera informacji o wielkości liter;
- cyfry i liczebniki – należy przeprowadzić normalizację (zamianę form pisanych na mówione, np. „1 " ⇒ „jeden") lub odwrotną normalizację (zamianę form mówionych na pisane, np. „jeden" ⇒ „1");
- znaki interpunkcyjne – te, które nie są werbalizowane oraz nie wpływają na sposób werbalizacji innych znaków lub wyrazów, można usunąć (na przykład wykrzyknik, znak zapytania, myślnik, kropka na końcu zdania). Pozostałe mogą być pozostawione lub razem z sąsiadującymi znakami zamienione na odpowiednią formę mówioną (np. „godz. 16:45" ⇒ „godzina szesnasta czterdzieści pięć").

[1] Są to słowa, których definicji zazwyczaj nie można znaleźć w słowniku i których pisownia nie przekłada się w bezpośredni sposób na wymowę. Przykład słów niestandardowych to cyfry i wyrażenia je zawierające, skróty. Dokładniejszy opis zagadnienia można znaleźć w pracy [129], która wprowadziła ten termin.

3.2.2. Metryki oceny modeli ASR

Naturalną kandydatem na metrykę służącą oceny modeli korekty jest metryka używana do oceny poprawianego systemu, lub metryki pochodne. Poniżej przedstawiono najpopularniejsze metryki służące do oceny wydajności systemów rozpoznawania mowy.

- Metryki operujące na poziomie jednego zdania:
  WER Word Error Rate – metryka powszechnie stosowana w ewaluacji systemów ASR oraz innych modeli zwracających tekst, na przykład modeli tłumaczenia maszynowego [61]. Przyjmuje wartości od 0.0 (w pełni poprawnie rozpoznane zdanie) do ∞. WER jest zdefiniowany jako:
  $$\frac{S + D + I}{N = C + S + D}$$
  gdzie:
  S to liczba zamienionych wyrazów (*substytucji*, ang. *Substitutions)*,
  D to liczba usuniętych wyrazów (*delecji*, ang. *Deletions*),
  I to liczba dodanych wyrazów (*insercji*, ang. *Insertions*),
  C to liczba prawidłowo rozpoznanych wyrazów,
  N to liczba wyrazów w zdaniu referencyjnym, równa sumie substytucji, delecji i prawidłowo rozpoznanych wyrazów.
  W celu uzyskania wartości poszczególnych składowych metryki (D, I, C, S) dokonuje się dopasowania (ang. *alignment*) sekwencji wyrazów z hipotezy i zdania referencyjnego za pomocą algorytmu minimalnej odległości edycyjnej [141]. Algorytm ten, używający programowania dynamicznego, znajduje najkrótszy ciąg operacji (insercji, delecji i substytucji) przekształcających jedną sekwencję w drugą.
  Pomimo powszechności stosowania WER jako metryki oceniającej systemy rozpoznawania mowy, nie zawsze dobrze oddaje ona jakość systemu rozpoznawania mowy w szerszym kontekście, na przykład całego systemu dialogowego, którego model ASR jest częścią [143].
  WRR Word Recognition Rate – zwana również „Word accuracy" albo W_{acc} - zdefiniowana jako:
  $$1 - WER$$
  - przyjmuje wartości od $-\infty$ (zazwyczaj jednak najmniejszą wartością jest 0) do 1.0. Wartość 1.0 oznacza całkowicie poprawnie rozpoznane zdanie.
- Metryki operujące na poziomie zbioru zdań:
  SER Sentence Error Rate[138] – stosunek liczby zdań niepoprawnie rozpoznanych (takich, których $WER > 0$) do liczby wszystkich zdań w zbiorze;
  SRR Sentence Recognition Rate, równiez "Sentence Accuracy" – stosunek liczby zdań w pełni poprawnie rozpoznanych (takich, których $WER == 0$) do liczby wszystkich zdań w zbiorze;
  WER Word Error Rate policzony dla całego zbioru zdań, po konkatenacji ich do jednego ciągu znaków;

WER_{avg} średnia wartość WER dla wszystkich zdań w zbiorze;
WER_{wavg} średnia WER dla wszystkich zdań w zbiorze, ważona ich długością.

3.2.3. Metryki używane do oceny modeli korekty błędów

Redukcja WER to różnica między wartością WER osiąganą przez system rozpoznawania przed i po zastosowaniu modelu korekty błędów. Może przyjmować wartości ujemne, co oznacza, że model korekty błędów pogorszył wynik WER całego systemu.

$$WER_{ASR} - WER_{Corrected} \tag{3.2}$$

gdzie:
WER_{ASR} to WER modelu ASR
$WER_{Corrected}$ to WER po zastosowaniu modelu korekty

Względna redukcja WER Opisana powyżej metryka *redukcji WER* nie nadaje się do porównywania modeli korekty błędów działających z różnymi modelami rozpoznawania mowy. Na przykład zredukowanie WER o 0.1 będzie znacznie łatwiejsze w przypadku modelu ASR osiągającego WER na poziomie 0.3 (redukcja z 0.3 do 0.2) niż w przypadku modelu osiągającego WER 0.1 (redukcja WER do poziomu 0.0, czyli poprawienie wszystkich błędów). Dlatego stosuje się [14][93] również metrykę *względnej redukcji WER*, daną wzorem:

$$\frac{WER_{ASR} - WER_{Corr}}{WER_{ASR}} \tag{3.3}$$

gdzie:
WER_{ASR} to WER modelu ASR,
WER_{Corr} to WER po zastosowaniu modelu korekty.
Jej prosta interpretacja to procent błędów ASR, które zostały naprawione.

3.2.4. Metryki zadań nadrzędnych

Jeśli system rozpoznawania mowy jest wykorzystywany jako część większego systemu, na przykład agenta dialogowego, metryką lepiej korelującą z zadowoleniem użytkownika i celami biznesowymi może być metryka oceniająca wpływ modelu zamiany mowy na tekst na metrykę oceniającą działanie całego systemu. Metryki specyficzne dla modeli ASR nie odróżniają od siebie błędów popełnianych na różnych wyrazach. Na przykład dla poprawnego zdania:

Zadzwoń na numer 123 456 789

i dwóch hipotez:

Zadzwoń na 123 456 789

Zadzwoń na numer 123 456 799

obie hipotezy osiągną tę samą wartość WER równą 0.167 (w pierwszym zdaniu jedna delecja w drugim jedna substytucja), ale pierwsza z nich doprowadzi do pomyślnego wykonania zadania przez agenta dialogowego, a druga może doprowadzić do fatalnej w skutkach pomyłki.

Dlatego pomiar skuteczności modeli korekty może opierać się na obliczeniu różnicy między wynikami osiąganymi przez model NLU na hipotezach ASR i hipotezach ASR poprawionych przez model korekty. Taką różnicę nazywamy zyskiem z korekty, np. zysk trafności klasyfikacji intencji może wynieść 10% jeśli trafność klasyfikacji przy użyciu hipotezy ASR wynosi 80% a przy zastosowaniu poprawionej hipotezy 70%.

Metryki powszechnie stosowane do mierzenia jakości modeli NLU to[33]:

- trafność klasyfikacji domeny,
- trafność klasyfikacji zamiaru,
- metryki oceniające poprawność wartości slotów, na przykład miara F1,
- EMA – z ang. *Exact Match Accuracy* – stosunek liczby przypadków w pełni poprawnie rozpoznanych przez model NLU (czyli, na przykład tych, których domena, zamiar i sloty zostały poprawnie rozpoznane) do liczby wszystkich testowanych przypadków.

Poza metrykami badającymi poszczególne wartości zwracane przez model NLU istnieją bardziej ogólne metryki oceniające działanie całego systemu dialogowego. Przykładem może być *wskaźnik powodzenia zadania* (ang. *task success rate*). Jest to miara, która określa jaki odsetek interakcji z agentem dialogowym zakończył się sukcesem, przy czym sukces jest określany zero-jedynkowo, jako zrealizowanie przez agenta czynności, o którą prosił użytkownik. Jest to miara, która z dotąd opisanych najlepiej oddaje cele biznesowe stojące za rozwojem systemów dialogowych, w tym systemu rozpoznawania mowy. Jej wykorzystanie jest jednak bardziej pracochłonne, wymaga bowiem dysponowania całym systemem dialogowym jak i odpowiednim zbiorem danych.

3.2.5. Metryki używane do oceny modeli tagowania

Przy ewaluacji modeli tagowania używa się następujących metryk, liczonych osobno dla każdej klasy (etykiety):

TP z ang. *True Positive* – liczba poprawnie otagowanych przypadków z danej klasy,

TN z ang. *True Negative* – liczba poprawnie otagowanych przypadków, które zostały otagowane jako nienależące do danej klasy,

FN z ang. *False Negative* – liczba przypadków niepoprawnie otagowanych jako nienależące do danej klasy,

FN z ang. *False Negative* – liczba przypadków niepoprawnie otagowanych jako należące do danej klasy,

Precyzja ang. *Precision* – dana wzorem

$$P = \frac{TP}{TP + FS}$$

to stosunek poprawnie zaklasyfikowanych przypadków z danej klasy do liczby wszystkich przypadków zaklasyfikowanych jako ta klasa,

Wrażliwość ang. *Recall* – dana wzorem

$$R = \frac{TP}{TP + FP}$$

to stosunek poprawnie zaklasyfikowanych przypadków z danej klasy do liczby wszystkich przypadków z tej klasy w rozpatrywanym zbiorze,

Miara F1 ang. *F-score* albo *F-measure* – miara łącząca wrażliwość i precyzję za pomocą średniej harmonicznej, dana wzorem:

$$F1 = 2 \times \frac{P \times R}{P + R}$$

W celu obliczenia metryk dla całego zbioru oblicza się średnią (ważoną liczbą przypadków, albo zwykłą) ostatnich trzech metryk dla wszystkich klas.

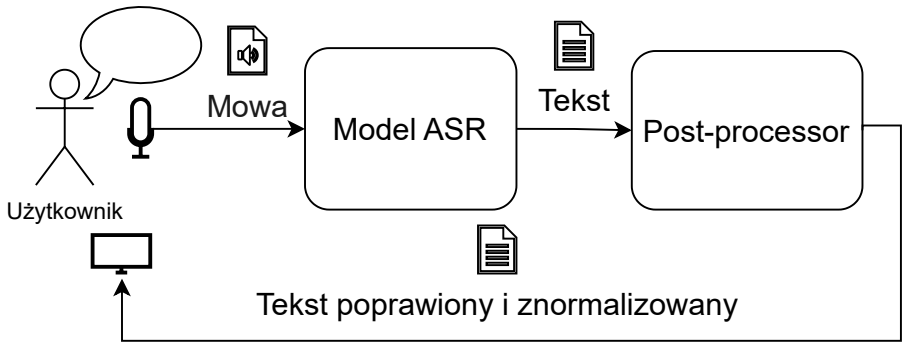
Rozdział 4

Modele automatycznej korekty błędów rozpoznawania mowy

Jak pokazano w rozdziale 1, problem rozpoznawania mowy nie jest problemem rozwiązanym i systemy zamiany mowy na tekst wciąż wymagają doskonalenia i adaptacji do zmieniających się warunków. W praktyce biznesowej podchodzi się do tego problemu na dwa sposoby. Pierwszy to bezpośrednia ingerencja w model rozpoznawania mowy, na przykład poprzez manipulację danymi trenującymi czy zmianę architektury. Takie podejście jednak wiąże się z pewnymi trudnościami:

- Trenowanie i adaptacja modeli ASR jest kosztowna obliczeniowo i zajmuje dużo czasu [106].
- Trenowanie modelu rozpoznawania mowy z użyciem nowych danych może przynieść poprawę wyników na tych danych, kosztem pogorszenia wyników w odniesieniu do uprzednio wykorzystywanych danych (zjawisko znane jako "Catastrophic forgetting" [92][34]).
- Nie jest możliwe w przypadku korzystania z gotowych, zamkniętych rozwiązań, na przykład kiedy używamy usługi rozpoznawania mowy działającej "w chmurze".

Drugi sposób, w praktyce stosowany jako uzupełnienie albo alternatywa do bezpośredniej zmiany modelu, to dokonanie adaptacji w module post-processor, który przetwarza tekst wychodzący z modelu rozpoznawania mowy [84][32][125](patrz rysunek 4.1).



Rysunek 4.1: Post-processor ASR

Takie podejście ma liczne zalety:

- Umożliwia szybkie wprowadzenie zmian, w porównaniu z adaptacją całego modelu.

- Umożliwia precyzyjną korektę poszczególnych błędów bez negatywnego wpływu na pozostałe przypadki.
- Może być zastosowane niezależnie od użytego modelu ASR, nawet jeśli nie występuje możliwości ingerencji w ten model.
- Nie wyklucza jednoczesnego stosowania adaptacji modelu ASR.

W praktyce biznesowej post-processor ASR najczęściej realizuje dwie funkcje [79]: dokonuje odwrotnej normalizacji tekstu [136] oraz poprawia błędy [43][77] rozpoznawania mowy wprowadzone przez model ASR.

Korekta błędów na etapie postprocessingu może być realizowana w różny sposób. Najprostsze jest podejście regułowe. W podejściu tym stosuje się zestaw reguł podstawieniowych, które za pomocą pewnego formalizmu, na przykład wyrażeń regularnych, definiują wzorce wyszukiwane w tekście wejściowym oraz odpowiadające im docelowe ciągi znaków podstawiane w miejsce ciągów znalezionych w tekście wejściowym. Przykładowe reguły można zobaczyć na listingu 4.1. Przykładowa reguła zawarta w drugiej linijce zamienia rozpoczynające zdanie ciągi znaków "co ja patrzę", "co ja paczę" lub "co ja patrze" na popularny w memach, choć niepoprawny ortograficznie ciąg "co ja pacze". Podejście to zapewnia precyzyjną kontrolę nad wynikami, wymaga jednak ręcznego tworzenia reguł. Tworzenie i utrzymywanie zbioru takich reguł wymaga znacznego nakładu pracy i jest do niego niezbędna zarówno biegłość techniczna (znajomość wyrażeń regularnych) jak i znajomość języka naturalnego, dla którego opracowywane są reguły. Dodatkowo zbiór reguł może wymagać częstej aktualizacji, wraz ze zmieniającym się modelem ASR, który generuje dane wejściowe do post-processora. Komplikuje to znacznie proces ręcznego utrzymywania i rozwoju reguł.

Listing 4.1: Przykład reguł post-processora. Lewa kolumna zawiera wyrażenie regularne, prawa podstawienie

```
pa trzy                          patrzy
^co ja pa(trz|cz)[ęe]            co ja pacze
a e?s e?r                        ASR
```

Opisane powyżej problemy z podejściem regułowym można rozwiązać za pomocą modeli automatycznej korekty błędów, które na podstawie błędów popełnianych przez konkretny system rozpoznawania mowy są w stanie nauczyć się poprawiać część z nich.

Bieżący rozdział przedstawia opis zaproponowanego przez autora podejścia "Otaguj i popraw", które pozwala w automatyczny sposób naprawiać błędy systemu rozpoznawania mowy na podstawie błędów popełnianych w przeszłości. Prace nad metodą automatycznej korekty błędów były rozłożone w czasie i wpisywały się w postępy dokonywane w tej dziedzinie. Korzystały z nowatorskich w trakcie ich prowadzenia metod uczenia maszynowego.

Zaproponowana metoda została wdrożona w przemysłowym systemie dialogowym firmy Samsung. Badania prowadzone nad modelami korekty zaowocowały powstaniem 3 publikacji naukowych dotyczących tego zagadnienia, spośród

których 2 powstały w całości w wyniku samodzielnej pracy autora [155][157] a jedna razem ze współautorami [72]. Metoda "Otaguj i popraw" została użyta do stworzenia rozwiązania w konkursie Poleval 2020, zadanie 1. "Post--editing and rescoring of automatic speech recognition results". Rozwiązanie to zajęło drugie miejsce w konkursie, nieznacznie tylko ustępując zwycięskiemu rozwiązaniu. Badanie nad modelami korekty błędów zaowocowały również opublikowaniem zbioru danych umożliwiających rozwój i ewaluację takich modeli [72].

Struktura rozdziału przedstawia się następująco: sekcja 4.1 stanowi przegląd literatury opisującej dotychczasowe rozwiązania problemu korekty błędów ASR. Sekcja 4.2 przedstawia opis proponowanej metody korekty błędów "Otaguj i popraw". W sekcji 4.3 opisano dane wykorzystane w przeprowadzonych badaniach oraz proces ich tworzenia. Sekcja 4.4 stanowi prezentację eksperymentów przeprowadzonych z użyciem zaproponowanej metody przy użyciu tych danych.

4.1. Przegląd literatury

Kompleksowy, choć już nie w pełni aktualny, przegląd metod korekcji błędów rozpoznawania mowy przedstawia praca [31], opisująca też metryki oceny systemów ASR. Cucu i in. [25] proponują korekcję błędów przy użyciu modelu SMT (Statistical Machine Translation). Model SMT jest trenowany na stosunkowo niewielkim równoległym korpusie 2000 transkrypcji ASR i ich ręcznie poprawionych wersji. W czasie oceny model jest używany do "tłumaczenia" hipotezy ASR do jej poprawionej formy. System osiąga 10.5% względnej redukcji WER zmniejszając WER bazowego systemu ASR z 11.4 do 10.2.

Przykładem podejścia sequence-to-sequence do korekty błędów jest rozwiązanie zaproponowane w pracy [43]. Autorzy podchodzą do problemu korekty jak do problemu tłumaczenia maszynowego: model uczy się „tłumaczyć" hipotezy na transkrypcje referencyjne. Taka metoda wymaga użycia dużych, jak na realia korpusów mowy, ilości danych trenujących. Dlatego autorzy wygenerowali dane za pomocą metody back-transcription (zob. rozdz. 6.2.1), polegającej na wygenerowaniu z danych tekstowych sygnału mowy za pomocą modelu syntezy mowy (ang. *TTS*) a następnie przetworzeniu tego sygnału z powrotem na tekst przez model ASR, dla którego tworzony jest model korekty błędów. Przy jej użyciu przetworzono 40 milionów zdań z tekstowej wersji korpusu LibriSpeech. Dla każdego zdania model ASR zwrócił listę 8 najbardziej prawdopodobnych hipotez (ang. *n-best list*), co razem dało 320 milionów par hipoteza-referencyjna transkrypcja. Wytrenowany przeez autorów model tłumaczenia to sieć typu encode-decoder z warstwami LSTM[50]. W pracy tej Guo i in. eksperymentowali również z poprawą wyników całego systemu ASR poprzez dołączenie do niego zewnętrznego modelu językowego oraz z połączeniem tych dwóch podejść. Proponowany system osiąga dobre wyniki (19% względnej redukcji WER i 29% względnej redukcji WER

z użyciem zewnętrznego modelu językowego, przy bazowym WER systemu ASR wynoszącym 6.03. Autorzy niestety nie podają opóźnień wprowadzanych przez model, ale wyniki podawane dla innych metod sequence-to-sequence sugerują, że mogą one być zbyt duże do zastosowania metody z systemami ASR działającymi w czasie rzeczywistym [86].

W pracy [77] przedstawiono model korekcji błędów dla języka mandaryńskiego nazwany "FastCorrect". Autorzy podkreślają znaczenie niskiego opóźnienia (latencji) modelu korekcji błędów ASR w środowiskach produkcyjnych. Zauważają, że większość modeli korekty błędów ASR używa autoregresywnego podejścia sequence-to-sequence, które pozwala osiągnąć znaczącą redukcję WER, ale za cenę wysokiej latencji. Dlatego proponują model, który nie jest autoregresywny. Co interesujące, autorzy początkowo prowadzili eksperymenty z zastosowaniem nieautoregresywanych modeli tłumaczenia maszynowego bezpośrednio do problemu korekty błędów, ale okazało się, że modele te zwiększały liczbę błędów, zamiast ją redukować. Zaproponowany model składa się z enkodera typu transformer, połączonego przez mechanizm atencji z dekoderem typu transformer. Dodatkowo, na wyjściach z warstwy ukrytej enkodera trenowana jest sieć przewidująca ilość tokenów wyjściowych odpowiadających danemu tokenowi wejściowemu. Dzięki informacji z tej sieci, w trakcie dekodowania poprawionego ciągu posiada dodatkową wskazówkę pozawalającą osiągnąć lepsze wyniki. Jako danych trenujących autorzy używają syntetycznyego korpusu par zdań poprawnych i niepoprawnych, generowanego przez losowe usuwanie, wstawianie i zastępowanie (z użyciem słów w zdaniach pochodzących z korpusu tekstowego. Rzeczywisty zestaw danych korekt ASR służy do dostrojenia modelu do konkretnego systemu ASR. Względna redukcja WER zaraportowana przez autorów na publicznie dostępnym zestawie testowym wynosi 13.87, co jest nieznacznie gorsze od modelu autoregresywnego (15.53), przy jednocześnie 6 razy niższej latencji (21 ms).

W pracy [86] Malmi i in. proponują metodę nazwaną "Lasser Tagger" korzystającą z tagowania operacjami edycyjnymi do rozwiązania zestawu różnych zadań NLP, takich jak tagowanie, łączenie i dzielenie zdań, tworzenie podsumowań i korekta błędów gramatycznych. W eksperymentach autorzy używają tagera opartego na architekturze transformer. Podobne rozwiązanie proponuje gu i in. w pacy [42]. Ich metoda przetwarza tekst poprzez stosowanie dwóch rodzajów operacji: delete i insert, na potrzeby zadań NLP takich jak tłumaczenie maszynowe, sumaryzacja czy post-edycja wyników tłumaczenia. Publikacja [85] opisuje warsztat poświęcony metodom generowania tekstu poprzez tagowanie, takim jak dwie wspomniane powyżej [86][42].

4.2. Metoda korekty błędów "Otaguj i popraw"

Badania nad modelami korekty błędów rozpoznawania mowy na etapie przetwarzania końcowego autor rozpoczął podczas prac nad modułem post-proces-

sora modułu ASR realizującego między innymi zadanie odwrotnej normalizacji tekstu (moduł ITN z angielskiego "Inverse Text Normalizer") oraz manualnej korekty błędów. Rozwój reguł normalizacji i korekty na potrzeby modułu był procesem wymagającym sporych nakładów pracy wysoko wykwalifikowanych pracowników, posiadających zarówno umiejętności techniczne, takie jak znajomość gramatyki Thrax [116] czy wyrażeń regularnych, jak i bardzo dobrą znajomość wybranego języka naturalnego. Iteracyjny proces rozwoju modeli ASR sprawia, że z każdą wersją modelu rozpoznawania mowy pojawia się potrzeba opracowania nowych reguł korekty błędów. Stąd pojawiła się potrzeba automatyzacji produkcji reguł korekty. Przegląd literatury przeprowadzony w tamtym czasie (około roku 2014) nie ujawnił istniejących rozwiązań problemu automatycznej korekty błędów w systemach ASR. Przystąpiono więc do prac nad własnym rozwiązaniem.

Początkowo przyjęte podejście polegało na indukcji reguł przepisywania w postaci: $(wzorzec, podstawienie)$, gdzie wzorzec stanowiło proste wyrażenie regularne, akceptujące jeden lub kilka wyrazów a podstawienie było ciągiem poprawnych wyrazów. Aby wyindukować zbiór takich reguł na danym korpusie poprawek użyto następującego algorytmu:

1. Dla każdej pary zdań hipoteza – zdanie referencyjne:
   a) dopasuj ciągi, otrzymując odpowiadające sobie fragmenty zdań,
   b) dodaj do zbioru różnic wszystkie pary, których fragmenty nie są identyczne.
2. Usuń ze zbioru sprzeczne pary, czyli takie, które mają identyczny fragment hipotezy a różny fragment zdania referencyjnego.

Tak otrzymany zbiór par (fragment hipotezy, fragment zdania referencyjnego) mógł zostać użyty do korekty pewnej klasy błędów. Podejście takie jednak okazało się obarczone wadami:

- Nie uwzględniało kontekstu zamienianych ciągów, przez co część reguł była odrzucana na etapie usuwania sprzecznych par, a część okazywała się mieć zbyt szeroki zakres, prowadząc do błędów fałszywie dodatnich, czyli zamiany fragmentów prawidłowych na nieprawidłowe.
- Nie generalizowało się na przypadki spoza zbioru trenującego.

Wymienione powyżej problemy z opisanym podejściem skłoniły do poszukiwania rozwiązania, które będzie ich pozbawione. Jego opis znajduje się poniżej.

4.2.1. Ogólny zarys metody

Problem korekty błędów w dowolnym tekście można ująć na jeden z dwóch sposobów. Pierwszy to potraktowanie go bezpośrednio jako problemu zamiany jednego tekstu na drugi, przy pomocy modeli sequence-to-sequence trenowanych w metodologii end-to-end (e2e), na podobieństwo modeli tłumaczenia maszynowego. Model taki (najczęściej rekurencyjna sieć neuronowa

[134][16][5] albo sieć typu transformer [140]) otrzymuje na wejściu tekst, zamienia go na wewnętrzną reprezentację sieci za pomocą enkodera, która następnie jest zamieniana na docelowy tekst za pomocą dekodera. Przegląd literatury pozwala wyciągnąć wniosek, że przeważająca większość modeli korekty błędów stosuje właśnie podejście e2e. Podejście to można określić jako oparte na danych (ang *data-driven*) – tworzenie modeli w tym paradygmacie nie wymaga od człowieka specjalistycznej wiedzy dotyczącej dziedziny (w tym przypadku błędów systemu rozpoznawania mowy i znajomości danego języka naturalnego), nie wymaga też tworzenia reguł. Wpływ na osiągane wyniki, przy ustalonej architekturze sieci, mają wyłącznie dane trenujące. Sprawia to, że podejście to łatwo zastosować do nowych dziedzin (na przykład do nowego modelu rozpoznawania mowy). Podejście end-to-end do korekty błędów ma kilka wad:

- Wymaga większej ilości danych trenujących [85][86] – w przypadku problemu korekty błędów ASR są one kosztowne (korpusy mowy wymagają ręcznej transkrypcji nagrań) [4][52][142][88]. Pewnym rozwiązaniem może być zastosowanie modelu syntezy mowy (TTS) do wygenerowania mowy z danych tekstowych i rozszerzenie za pomocą tak stworzonych danych korpusu mowy, który następnie zostaje zamieniony na tekst przez model ASR [43]. To rozwiązanie powoduje jednak wprowadzenie do danych trenujących błędów popełnianych podczas syntezy mowy przez model TTS i zwiększa złożoność obliczeniową całego rozwiązania.
- Jest rozwiązaniem typu black-box – nie umożliwia łatwej kontroli nad działaniem modelu [85][86]. Kontrola jest możliwa na etapie trenowania – przez dobór danych trenujących i parametrów trenowania sieci. Na etapie inferencji kontrola jest możliwa jedynie na podstawie prawdopodobieństw zwracanych przez model dla każdego słowa na wyjściu (na przykład jeśli średnie albo minimalne prawdopodobieństwo jest niższe niż pewien ustalony próg, to model zwraca oryginalne, niepoprawione zdanie).
- Modele neuronowe end-to-end są podatne na zjawisko halucynacji [86] [91] [105], polegające na zwracaniu na wyjściu ciągów wyrazów w żaden sposób niezwiązanych ze źródłowym zdaniem, czasami w postaci powtarzającego się wiele razy jednego wyrazu.
- Jest wymagające obliczeniowo podczas inferencji [86][85][42] przez co nie nadaje się do zastosowań gdzie kluczowy jest krótki czas wykonania, jak na przykład rozpoznawanie mowy na potrzeby agentów dialogowych.

Drugie podejście do problemu korekty błędów to potraktowanie go jako problemu tagowania sekwencji. Tagowanie sekwencji polega na przypisaniu każdemu jej elementowi (na przykład każdemu wyrazowi w zdaniu) pewnej etykiety. Przykładem zadania NLP polegającego na tagowaniu sekwencji jest zadanie przypisywania części mowy (ang. *POS-tagging*[1]).

Zaproponowana metoda korekty błędów została przez autora nazwana *tag and correct*, czyli "Otaguj i popraw", ponieważ działa w dwóch krokach:

[1] POS-tagging - Part Of Speech Tagging

otaguj – każdy token w rozpatrywanym tekście jest klasyfikowany jako poprawny lub zostaje mu przypisana etykieta identyfikująca błąd i informująca o tym, jak może on zostać poprawiony,
popraw – tokeny zaklasyfikowane jako błędne są naprawiane, korzystając z przypisanej im etykiety.

Aby przypisane do błędnie rozpoznanych wyrazów etykiety umożliwiały korektę błędów w jednoznaczny sposób, w zaproponowanym podejściu jako etykiety zostały użyte nazwy operacji edycyjnych. Określają one jak zamienić jeden ciąg wyrazów w drugi. Na przykład operacja edycyjna *dopisz na końcu „ch"* zamieni wyraz „łazienka" w wyraz „łazienkach".

Metoda "Otaguj i popraw" składa się z dwóch etapów:

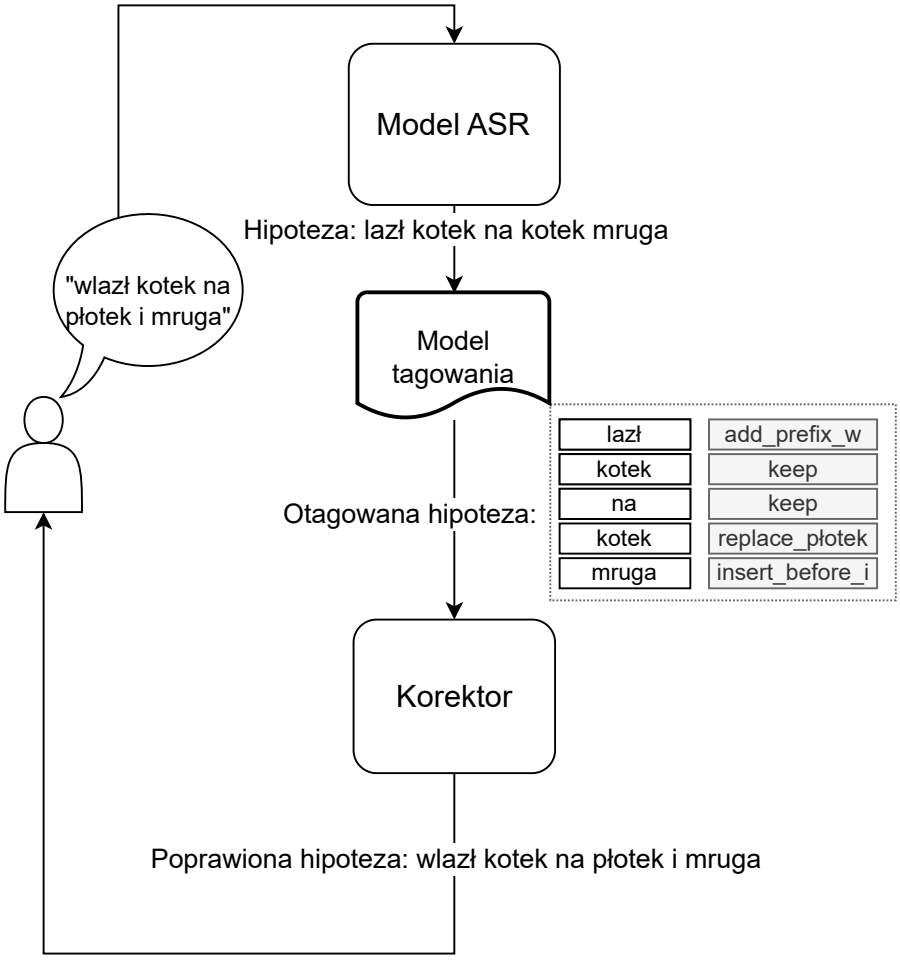
- Etap uczenia modelu:
  - Wykorzystując wejściowe dane, składające się z hipotez systemu ASR wraz z odpowiadającymi im referencyjnymi transkrypcjami, generowane są dane trenujące w postaci hipotez otagowanych operacjami edycyjnymi.
  - Wygenerowane w pierwszym kroku dane są używane do wytrenowania statystycznego modelu tagowania operacjami edycyjnymi.
- Etap wykorzystania modelu:
  - Hipoteza systemu rozpoznawania mowy zostaje otagowana przez wytrenowany model – krok "otaguj".
  - Operacje edycyjne są aplikowane przez *korektor* do otagowanych nią hipotez – krok "napraw".

Przykład działania metody podczas używania modelu do korekty błędów przedstawiono na rysunku 4.2. Widoczne są na nim następujące kroki:

1. Użytkownik wypowiada do systemu ASR zdanie "wlazł kotek na płotek i mruga",
2. Model rozpoznawania mowy popełnia błąd zwracając hipotezę "lazł kotek na kotek mruga",
3. Model tagowania przypisuje wyrazom w hipotezie etykiety operacji edycyjnych: lazł`[add_prefix_w]` kotek`[keep]` na`[keep]` kotek`[replace_płotek]` mruga`[insert_before_i]`,
4. Korektor aplikuje operacje do wyrazów w hipotezie zwracając poprawione zdanie "wlazł kotek na płotek i mruga" do użytkownika.

Na metodę składają się następujące elementy:

- zaproponowany przez autora zbiór szablonów operacji edycyjnych dla problemu korekty błędów ASR (sekcja 4.2.2),
- zaproponowana przez autora metoda generowania operacji edycyjnych (sekcja 4.2.2),
- wybrana dla konkretnego problemu metoda trenowania modeli tagowania. Sekcja 4.4 opisuje eksperymenty przeprowadzone z różnymi modelami tagowania,



Rysunek 4.2: Metoda "Otaguj i popraw" użyta do korekty błędu ASR

- zaproponowana przez autora metoda aplikowania operacji edycyjnych przez *korektor*, umożliwiająca precyzyjną kontrolę nad działaniem mechanizmu korekty, opisana w sekcji 4.2.5.

4.2.2. Zbiór operacji edycyjnych

Dobór zbioru operacji edycyjnych powinien zależeć od rozwiązywanego problemu, czyli od tego, jakie błędy najczęściej model wytrenowany przy ich użyciu będzie naprawiał. Z jednej strony, projektując taki zbiór, powinno się dążyć do jego minimalizacji. Dzięki temu model tagowania będzie łatwiej wytrenować, a dane trenujące będą zawierały mniej rzadkich etykiet, na przykład takich, które w całym zbiorze trenującym występują tylko raz. Wymagamy więc od tego zbioru by był *specyficzny* dla danego problemu. Jednocześnie zbiór ten musi umożliwiać wyrażenie wszystkich możliwych różnic między dwoma ciągami znaków, powinien być więc *uniwersalny*.

Ręczne zaprojektowanie zbioru wszsytkich operacji edycyjnych jest praktycz-

nie niewykonalne,[2] ale można określić szablony (lub *klasy*) operacji, które potem służą do stworzenia konkretnych operacji przy użyciu wybranego zbioru poprawek. Na przykład, szablonem operacji może być `append_{}` a konkretną operacją `append_ch` – operacja, która dodaje do końca wyrazu znaki „ch", na przykład zamieniając wyraz „łazienka" w „łazienkach". Dobór i definicja szablonów operacji edycyjnych jest zadaniem, które musi wykonać człowiek, opierając sie na swoim doświadczeniu oraz analizując błędy popełniane przez modele rozpoznawania mowy. Konkretny zbiór operacji edycyjnych powstaje poprzez zastosowanie zaprojektowanych ręcznie szablonów operacji edycyjnych do danych trenujących.

Operacje edycyjne podzielono na trzy kategorie:

1. operujące na całych wyrazach – na przykład zamieniające jeden wyraz w drugi,
2. operujące na znakach w obrębie wyrazu, na przykład dopisujące końcówkę do istniejącego wyrazu,
3. łączące/dzielące wyrazy – zmieniające liczbę wyrazów w zdaniu poprzez operowanie spacjami.

Tabela 4.1 prezentuje wszystkie klasy (szablony) operacji edycyjnych zaproponowane przez autora i używane w omawianych modelach korekty błędów, wraz z przykładami konkretnych operacji oraz przykładami ich zastosowania. Dla operacji `keep`, `del` i `join` szablon jest tożsamy z konkretną operacją. W przypadku pozostałych szablonów, operacja powstaje poprzez podstawienie w miejsce nawiasów klamrowych wartości (znaku alfanumerycznego albo ciągu znaków).

Zaproponowany przez autora zbiór szablonów operacji edycyjnych powstał w wyniku analizy najczęstszych błędów popełnianych przez system ASR, z myślą o którym powstawała metoda "Otaguj i popraw". Jednocześnie był projektowany tak, żeby być uniwersalnym, czyli żeby dla każdego potencjalnego modelu ASR i korpusu mowy generować możliwie jak najmniejszy zbiór operacji pokrywających wszystkie błędy popełniane przez model ASR na tym korpusie.

Przykładem odwrotnego podejścia do konstrukcji zbioru operacji edycyjnych jest metoda edycji tekstu LasserTagger [86] zastosowana do szerszej klasy problemów (patrz. rozdział 4.1). Zastosowany w niej zbiór szablonów ma tylko dwa elementy: `KEEP` i `DELETE`. Konkretna operacja edycyjna powstaje poprzez dodanie do szablonu słowa (może być puste) ze słownika, które ma być wstawione przed otagowanym wyrazem. W rezultacie otrzymujemy odpowiedniki szablonów operacji z metody "Otaguj i popraw":

`keep` – szablon `KEEP` z pustym słowem

[2] jeśli taki zbiór ma pokrywać wszytkie możliwe błędy, to dla samych tylko błędów zamiany (substytucji) musiałby być w stanie wyrazić $|V|^2 - |V|$ zamian, gdzie $|V|$ to wielkość słownika używanego przez model ASR (liczba różnych słów, które może zwrócić model ASR)

`del` – szablon `DELETE` z pustym słowem
`insert_before_{}` – szablon `KEEP` z niepustym słowem
`replace_{}` – szablon `DELETE` z niepustym słowem

Zbiór operacji edycyjnych powstały przez zastosowanie jedynie tych czterech szablonów jest wystarczający, żeby odwzorować wszystkie możliwe różnice między hipotezami i zdaniami referencyjnymi (jest *uniwersalny*), jednak jest niepotrzebnie duży (nie jest *specyficzny* dla danego problemu).

Na przykład, dla języka angielskiego, każdy błąd polegający na użyciu formy pojedynczej, zamiast mnogiej musiałby być wyrażony osobną operacją edycyjną:

- cat $\xrightarrow{\text{replace\_cats}}$ cats
- dog $\xrightarrow{\text{replace\_dogs}}$ dogs
- bird $\xrightarrow{\text{replace\_birds}}$ birds

...

Dziękki użyciu w metodzie "Otaguj i popraw" operacji edycyjnych na poziomie znaków (a nie całych wyrazów) wszystkie te błędy można wyrazić za pomocą jednej operacji `add_suffix_s`:

- cat $\xrightarrow{\text{add\_suffix\_s}}$ cats
- dog $\xrightarrow{\text{add\_suffix\_s}}$ dogs
- bird $\xrightarrow{\text{add\_suffix\_s}}$ birds

...

Model tagowania może dzięki użyciu takiego zbioru operacji nauczyć się generalizować widziane przypadki na niewdziane do tej pory (na przykład ucząc się reguły "kiedy po rzeczowniku zakończonym na inną literę niż »s« występuje wyraz »are«, otaguj ten rzeczownik operacją `add_suffix_s`").

4.2.3. Generowanie referencyjnych operacji edycyjnych

Wytrenowanie modelu tagującego zdania operacjami edycyjnymi wymaga przygotowania zbioru trenującego, zawierającego zdania z błędami (hipotezy ASR), w których każdy wyraz ma przypisaną oczekiwaną etykietę operacji edycyjnej. W metodzie "Otaguj i popraw" etykiety są generowane z korpusu poprawek, czyli par hipoteza ASR–poprawna transkrypcja. Dla każdej takiej pary przeprowadzana jest normalizacja obu zdań, tak żeby operacje edycyjne dotyczyły tylko tych różnic, które tworzony model ma poprawiać. Rodzaje i sposoby normalizacji są takie same jak podczas normalizacji na potrzeby ewaluacji, opisanej w sekcji 3.2. Zastosowanie konkretnych zasad normalizacji zależy od wymagań stawianych modelowi (na przykład czy oprócz korekty błędów ma również dokonywać normalizacji). W prezentowanej metodzie przyjęto rozwiązanie, że znormalizowane zdania są dopasowywane i porównywane za pomocą algorytmu Ratcliffa-Obershelpa [58] zaimplementowanego w bibliotece difflib [137]. Algorytm ten szuka najdłuższego wspólnego podciągu w dwóch

szablon	**opis**	**przykład**
	na całych wyrazach	
`keep`	zachowaj wyraz bez zmian	"kotek" $\xrightarrow{keep}$ "kotek"
`del`	usuń otagowany wyraz	"na" $\xrightarrow{del}$ ""
`replace_{w}`	zamień otago. wyraz na wyraz w	„kotek" $\xrightarrow{\text{replace\_płotek}}$ "płotek"
`insert_before_{w}`	wstaw wyraz w przed otag. wyrazem	„mruga" $\xrightarrow{\text{insert\_before\_i}}$ „i mruga"
`insert_after_{w}`	wstaw wyraz w po otag. wyrazie	"ładna" $\xrightarrow{\text{insert\_after\_to}}$ "ładna to"
	w obrębie wyrazu	
`add_prefix_{p}`	dopisz prefiks p na początku wyrazu	"lazł" $\xrightarrow{\text{add\_prefix\_w}}$ "wlazł"
`add_suffix_{s}`	dopisz suffiks s na końcu wyrazu	"zaśpiewa" $\xrightarrow{\text{add\_suffix\_j}}$ "zaśpiewaj"
`del_suffix_{n}`	usuń n znaków z końca wyrazu	"piosenkach" $\xrightarrow{\text{del\_suffix\_2}}$ "piosenka"
`del_prefix_{n}`	usuń n znaków z początku wyrazu	"oto" $\xrightarrow{\text{del\_prefix\_2}}$ "to"
`replace_suffix_{s}`	zamień ostatnie $len(s)$ znaków ciągiem s	"koteczek" $\xrightarrow{\text{replace\_suffix\_ku}}$ "koteczku"
`sreplace_{s}_{r}`	zamień podciąg s ciągiem r	"piosneczka" $\xrightarrow{sreplace\_necz\_en}$ „piosenka"
	podziel / złącz	
`join`	złącz wyraz z następnym	"nie długa" $\xrightarrow{\text{join}}$ "niedługa"
`join_{s}`	złącz wyraz z nast. używając ciągu s	"bielsko biała" $\xrightarrow{\text{join\_-}}$ "bielsko-biała"
`split_aftert_{n}`	podziel wyraz po n-tym znaku	"wsam" $\xrightarrow{\text{split\_aftert\_1}}$ "w sam"
`split_on_first_{c}`	podziel wyraz po pierwszym znaku c	"niezgadnie" $\xrightarrow{\text{split\_on\_first\_e}}$ "nie zgadnie"
`split_on_last_{c}`	podziel wyraz po ostatnim znaku c	"ryba-piła" $\xrightarrow{\text{split\_on\_last\_-}}$ "ryba piła"

Tabela 4.1: Klasy operacji edycyjnych wraz z przykładami konkretnych operacji, użyte w autorskim podejściu "Otaguj i popraw".

ciągach S_1 i S_2 a następnie działa rekurencyjnie dla niepasujących fragmentów po obu stronach znalezionego podciągu. Zwraca wartość w przedziale $(0, 1.0)$ obliczną według wzoru:

$$D_{ro} = \frac{2K_m}{|S_1| + |S_2|}$$

gdzie K_m to liczba zgodnych znaków w dwóch podciągach. Chociaż głównym celem algorytmu jest zwrócenie miary podobieństwa D_{ro} między ciągami, to efektem pobocznym jego działania jest zwrócenie odpowiadających sobie podciągów z S_1 i S_2 wraz z informacją, czy ciągi te są identyczne, czy różne. Dla dwóch porównywanych zdań S_1 i S_2 algorytm Ratcliffa-Obershelpa zwraca tę informację w postaci dwóch par indeksów (i_{S1}, i'_{S2}) i (i_{S2}, i'_{S2}) oznaczających pozycję fragmentów w obu ciągach oraz jednej z czterech etykiet operacji zamieniających jeden ciąg S_1 w ciąg S_2:

equal – podciągi są identyczne,
replace – zamień podciąg $S_1[i_{S1} : i'_{S1}]$ na $S_2[i_{S2} : i'_{S2}]$,
delete – usuń podciąg $S_1[i_{S1} : i'_{S1}]$,
insert – wstaw podciąg $S_2[i_{S2} : i'_{S2}]$ do S_1 w miejscu i_{S1} przesuwając znaki do przodu.

Zwracane przez algorytm Ratcliffa-Obershelpa etykiety nie są docelowymi operacjami edycyjnymi, którymi chcemy otagować hipotezę, ponieważ wymagają dostępu do docelowego (referencyjnego) zdania, żeby zamienić w nie hipotezę.

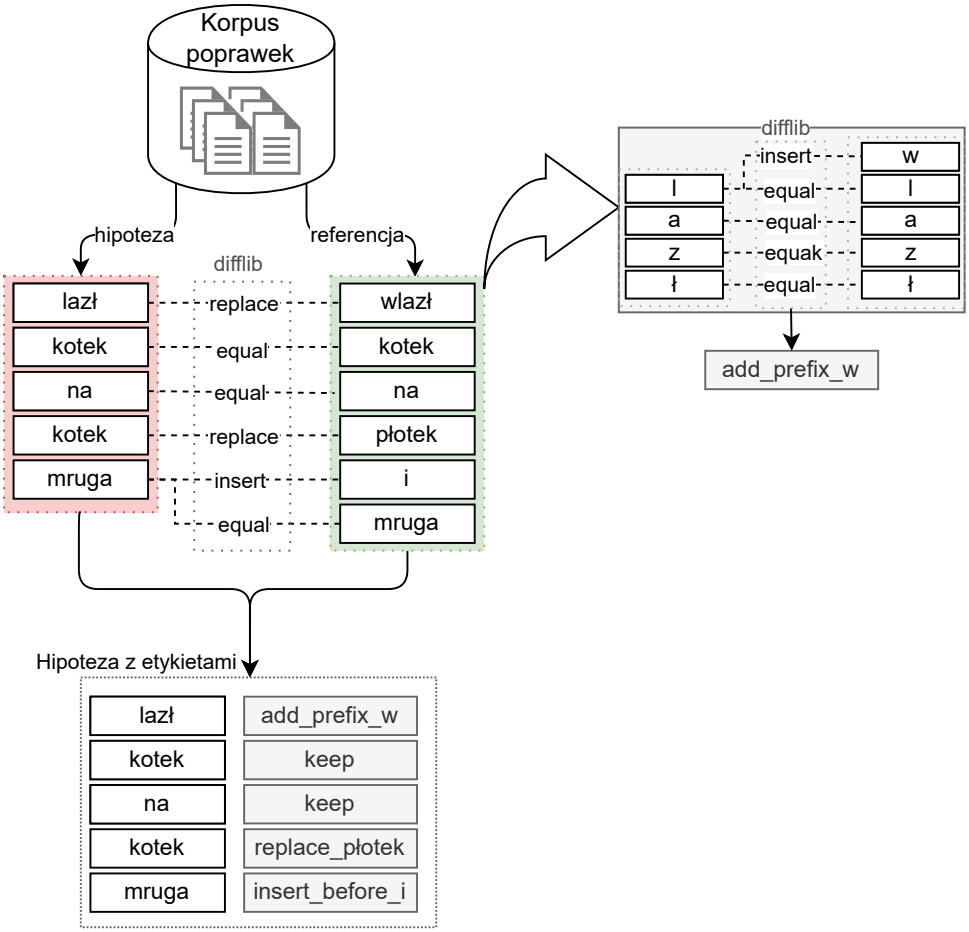
Rozwiązaniem tego problemu mogłoby być stworzenie operacji edycyjnych poprzez dopisanie do każdej etykiety ”replace” i ”insert” odpowiedniego podciągu znaków z docelowego zdania. W ten sposób jednak otrzymalibyśmy jedynie operacje edycyjne trzech pierwszych rodzajów z tabeli 4.1: `del`, `replace_{w}` oraz `insert_before_{w}`, czyli operacje operujące na całych wyrazach. Jak wykazano w poprzedniej sekcji, taki zbiór operacji byłby wystarczający, ale nieoptymalny dla problemu korekty błędów ASR.

Uzyskanie operacji edycyjnych na poziomie znaków oraz operacji łączenia-/rozdzielania wyrazów jest możliwe dzięki zastosowaniu algorytmu Ratcliffa-Obershelpa najpierw dla sekwencji wyrazów, a następnie dla różniących się fragmentów na poziomie znaków. Z pomocą operacji zwróconych przez difflib na poziomie znaków oraz porównując różniące się podciągi za pomocą kaskady ręcznie stworzonych reguł można wygenerować wszystkie operacje edycyjne wymienione w tabeli 4.1.

Niekiedy tę samą różnicę między ciągami wyrazów można wyrazić przy pomocy różnego zestawu operacji edycyjnych. W takim przypadku algorytm preferuje operacje, które będą się lepiej generalizować i na przykład operacja `replace_{w}` jest zwracana w ostateczności, kiedy różnic nie da się wyrazić za pomocą operacji wewnątrz wyrazu.

Przykład generowania tagów z pary zdań pokazano na rysunku 4.3. Wejście do procesu stanowią pary hipoteza ASR (potencjalnie błędna) i zdanie referencyjne. Sekwencje wyrazów są porównane za pomocą algorytmu Ratcliffa-Obershelpa. Dla pierwszej pary odpowiadających sobie wyrazów ”lazł” i ”wlazł” porównanie sekwencji wyrazów zwraca etykietę `replace`. Analiza różnic na poziomie sekwencji znaków prowadzi do przypisania wyrazowi ”lazł” operacji `add_prefix_w`. Dwa kolejne wyrazy w porównywanych zdaniach są identyczne, mają więc przypisaną operację `keep`. Różnice między wyrazami ”kotek” i ”płotek” nie podpadają pod schemat żadnej operacji na znakach, wyrazowi ”kotek” zostaje więc przypisana operacja `replace_with_płotek`. Etykieta ”insert” przypisana przez algorytm Ratcliffa-Obershelpa występującemu w zdaniu referencyjnym wyrazowi ”i” przekłada się na operację edycyjną `insert_before_i` przypisaną następującemu po nim wyrazowi ”mruga”.

Wydajność trenowania modeli tagowania spada wraz z rosnącą wielkością zbioru tagów, a jednocześnie tagi rzadko występujące w danych trenujących są przypisywane przez tager z niższą jakością. Dlatego w ramach stosowanej metody narzuca się górne ograniczenie na rozmiar zbioru operacji edycyjnych znajdujących się w zbiorze trenującym tager. W tym celu, po wygenerowaniu danych trenujących, a przed wytrenowaniem modelu, zachowuje się N najczęściej występujących w tym zbiorze operacji edycyjnych a pozostałe, mniej liczne operacje zastępuje się etykietą `UNSOPPORTED_OPERATION`, która podczas aplikowania operacji edycyjnych jest traktowana tak samo jak `keep`.



Rysunek 4.3: Przykład generowania tagów z korpusu poprawek

4.2.4. Modele tagowania

Wygenerowany zgodnie z opisem w poprzednich sekcjach zbiór danych, składający się z błędnych zdań otagowanych operacjami edycyjnymi, służy do wytrenowania modelu tagowania. Każda metoda tagowania może być użyta w tym celu. Podczas prac badawczych przetestowano metody przedstawione poniżej. Ich dokładny opis znajduje się w rozdziale 3.1.

1. Tager Brilla – prosty tager regułowy. W zaprezentowanych eksperymentach jest stosowany jako rozwiązanie bazowe, za założeniem, że bardziej wyrafinowane metody powinny osiągnąć znacznie lepsze wyniki.
2. BERT – modele tagowania wykorzystujące neuronowy model języka w architekturze Transformer [140]. W momencie przeprowadzania eksperymentów tagery oparte na modelu BERT osiągały wyniki zbliżone do ówczesnych wyników state-of-the-art. Jednocześnie model BERT jest otwarcie dostępny, ma swoje wersje dla języków naturalnych, na których przeprowadzano eksperymenty i jak pokazują wyniki eksperymentów, jest mniej wymagający obliczeniowo od bardziej złożonego modelu Flair.
   W opisywanych w tej pracy modelach tagowania pretrenowane modele

typu BERT, z dodaną jedną warstwą wyjściową, były dotrenowywane do problemu tagowania sekwencji (patrz opis procedury w sekcji 3.1)

3. Tager "Flair" – model tagowania zaimplementowany w oparciu o bibliotekę NLP Flair [1]. Został on wybrany ze względu na raportowane wyniki state-of-the-art w zadaniach tagowania częściami mowy (*POS tagging*) [3] oraz encjami nazwanymi (*NER tagging*) [10][2][3]. W opisywanych w niniejszej pracy modelach opisanych jako „Flair", użyty został następująca architektura, wzorowana na osiągających wyniki SOTA tagerach NER [132][126]:
   a) wektoryzacja wejścia za pomocą *Contextual word embeddings* [3],
   b) pretrenowana sieć BERT [28] (w dwóch z trzech eksperymentów z użyciem Flair),
   c) warstwa LSTM [50],
   d) CRF (patrz sekcja 3.1)

4.2.5. Aplikowanie operacji edycyjnych

Za pomocą wytrenowanego modelu tagowania, każdemu wyrazowi w hipotezie pochodzącej z systemu rozpoznawania mowy zostaje przypisana operacja edycyjna. Aby naprawić błędy, operacje należy zastosować do wyrazów w hipotezie, modyfikując je. Odpowiedzialny za to moduł nazwany został *korektorem*.

Dzięki mechanizmowi aplikowania operacji edycyjnych metoda "Otaguj i popraw" umożliwia precyzyjną kontrolę zachowania modułu korekty już po wytrenowaniu modelu, na etapie inferencji, co nie jest możliwe w przypadku metod niekorzystających z operacji edycyjnych [43][77].

Kontrola zachowanie korektora jest możliwa na kilku poziomach. Na najbardziej gruboziarnistym poziomie, możemy skorzystać z wartości oceny zwracanej przez tager. Podczas inferencji tager przypisuje każdemu wyrazowi w wejściowym zdaniu etykietę operacji edycyjnej. W przypadku tagerów stochastycznych model zwraca również wartość oceny zaufania (ang. *confidence score*) – wartość rzeczywistą z przedziału $[0; 1.0]$, oznaczającą prawdopodobieństwo, że przypisanie jest prawidłowe (nie musi być to prawdopodobieństwo w ścisłym znaczeniu, ale wartości dla wszystkich etykiet powinny sumować się do 1.0). Wartość zwróconą przez tager można wykorzystać w celu sterowania precyzją systemu korekty. Ustawiając dla niej próg, możemy odfiltrować wszystkie operacje, których tager nie jest wystarczająco „pewny", zwiększając tym samym jego precyzję kosztem czułości (*recall*).

Nieco dokładniejszą metodą odfiltrowania operacji edycyjnych na etapie ich aplikowania jest użycie wyników ewaluacji tagera. Ewaluacja samego modelu tagera, w oderwaniu od reszty metody korekty błędów, polega na otagowaniu przy jego pomocy tekstu, dla którego znamy poprawne tagowania i porównaniu oczekiwanych tagów ze zwróconymi. Wyniki tego porównania oblicza się przy użyciu metryk używanych przy ewaluacji tagerów takich jak specyficzność (albo *precyzja*, ang. *precission*, wrażliwość(ang. *recall*) i miara F1 (zobacz rozdzial 3.2.5). Po dokonaniu ewaluacji tagera na zbiorze ewaluacyjnym można

wykluczyć z wykonywania operacje edycyjne, które na zbiorze testowym osiągnęły niską wartość tych miar, w szczególności niską precyzję. Niska precyzja dla danej etykiety oznacza, że tager często otagowuje tą operacją wyrazy, które nie powinny być nią otagowane. W przypadku operacji edycyjnych oznacza to wprowadzanie nowych błędów zamiast naprawiania istniejących. Niska wrażliwość jest znacznie mniej szkodliwa – oznacza, że dana operacja często nie zostanie wykonana i błąd nie zostanie naprawiony.

Najbardziej precyzyjną metodą filtrowania operacji edycyjnych jest samodzielne zdefiniowanie reguł filtrowania po dokonaniu inspekcji błędów popełnianych przez tager. Dzięki takim regułom na etapie inferencji można zrezygnować z wykonywania konkretnych reguł w danym kontekście (dla konkretnych słów), przy zadanym progu wartości oceny. Przykładowe reguły filtrowania operacji przedstawiono na listingu 4.2.

Listing 4.2: Przykład reguł filtrujących operacje edycyjne

```
{[
        [
            {"word":".+", "label":".+", "score":0.5}
        ],
        [
            {"word":".+", "label":"del", "score":0.9}
        ],
        [
            {"word":"^", "label":"^", "score":0},
            {"word":"czym", "label":"del_suffix_1",
               ↪ "score":0.9},
            {"word":"jest|żmona|ęmog", "label":"keep",
               ↪ "score":0}
        ],
        [
            {
            "word":".+",
            "label":"replace_kot|join_-|del_suffix_3",
            "score":0
            }
        ]
]}
```

W powyższym przykładzie zawarte są cztery reguły filtrowania. Pierwsza z reguł jest regułą pierwszego rodzaju, to znaczy operuje tylko na wartości oceny zwróconej przez tager. Druga i trzecia reguła są regułami ostatniego rodzaju, dodanymi ręcznie. Reguła druga oznacza, że operacja `del`, czyli usunięcie wyrazu, może być wykonana tylko, jeśli tager zwrócił dla niej wartość oceny równą 0.9 albo wyższą. Trzecia reguła dotyczy operacji `del_suffix_1`: zostanie wykonana tylko, jeśli wyraz nią otagowany to „czym", wartość oceny zwrócona dla tej operacji przez tager jest mniejsza niż 0.9, jest to pierwszy wyraz w zdaniu a wyraz następujący po nim to jeden z wyrazów

„jest", „można", „mogę", otagowany etykietą `keep` z dowolną wartością oceny. Ostatnia reguła wyłącza z wykonywania 3 operacje edycyjne. Taka reguła mogłaby zostać automatycznie dopisana do zbioru reguł na podstawie wyników ewaluacji tagera.

Jak widać na powyższym przykładzie, zastosowanie modelu korekty błędów opartego o operacje edycyjne umożliwia bardzo precyzyjną kontrolę nad zachowaniem modelu korekty, w tym manualną korektę błędów popełnianych przez sam tager. To sprawia, że metoda ta nadaje się do zastosowań produkcyjnych, gdzie konieczna jest możliwość korekty pojedynczych przypadków.

4.3. Dane

Stworzenie i rozwój modelu korekty błędów wymaga przygotowania odpowiednich danych. Należy podkreślić, że jakość i ilość danych mają ogromne znaczenie dla jakości uzyskiwanych modeli uczenia maszynowego [24][55][44][112]. To dane definiują problem, który ma rozwiązać tworzony model uczenia maszynowego.

Dane używane do trenowania i ewaluacji modeli powinny jak najlepiej odzwierciedlać dane, z którymi model uczenia maszynowego będzie używany po jego wdrożeniu. W przypadku modelu korekty błędów rozpoznawania mowy dane wejściowe to wyniki zwracane przez konkretny model zamiany mowy na tekst a oczekiwane dane wyjściowe to poprawne transkrypcje wypowiedzi.

Dane zwracane przez system ASR mogą mieć różną formę:

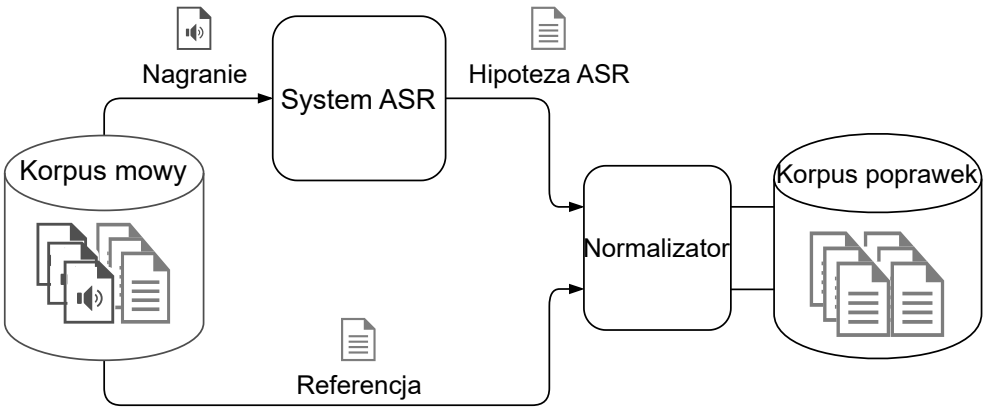
1. pojedyncza hipoteza systemu ASR (ang. *1-best hypothesis*), czyli ciąg znaków zawierający transkrypcję mowy,
2. lista hipotez (ang. *n-best list*) wraz z przypisaną do każdej hipotezy oceną (ang. *score*) informującą o tym jak model ocenia prawdopodobieństwo tego, że ta hipoteza jest poprawna,
3. krata (ang. *lattice*) - skierowany, acykliczny graf z wagami, w którym każda ścieżka od stanu początkowego do końcowego reprezentuje jedną hipotezę zwróconą przez ASR, przy czym sumaryczna waga całej ścieżki odpowiada ocenie hipotezy [82],
4. confusion network [87][75] – skierowany graf acykliczny, gdzie każda hipoteza musi przejść przez wszystkie węzły w grafie. Powstaje poprzez uliniowienie (ang. *alignment*) hipotez z kraty.

Oczekiwane dane wyjściowe mają standardowo formę pojedynczych wypowiedzi. W dalszej części pracy będziemy je nazywać "zdaniami referencyjnymi". Zdania te mogą mieć różne pochodzenie:

1. transkrypcja wypowiedzi stanowiących wejście do modelu ASR, dokonana przez człowieka odsłuchującego nagrania tych wypowiedzi,
2. zdanie, które osoba nagrywająca korpus mowy miała za zadanie wypowiedzieć.

Zauważmy, że zdania referencyjne pochodzące z wymienionych powyżej źródeł mogą różnić się ze względu na błędy popełniane przez ludzi podczas dokonywania nagrań do korpusu (przejęzyczenia, wtrącenia) oraz anotacji (literówki, nieświadome poprawianie błędów popełnionych przez osobę czytającą zdania w korpusie).

Do stworzenia powyższych danych potrzebny jest model rozpoznawania mowy, którego błędy chcemy poprawiać, oraz pary nagrana wypowiedź – transkrypcja. Schemat przygotowania korpusu poprawek, zawierającego pary "zdanie z błędami – poprawne zdanie", przedstawia rysunek 4.4.



Rysunek 4.4: Przygotowanie korpusu poprawek

Badania nad modelami korekty błędów opisywane w niniejszej pracy były przeprowadzone z wykorzystaniem zbiorów danych pochodzących z trzech źródeł:

- wewnętrzne korpusy mowy i wyniki ewaluacji systemu ASR firmy Samsung,
- wyzwanie „Post-editing and rescoring of automatic speech recognition results", Poleval 2020 [100],
- wyzwanie „Open Challenge for Correcting Errors of Speech Recognition Systems"(OCCESRS) [72] – dane przygotowane przez autora i współpracowników.

4.3.1. Korpusy mowy firmy Samsung

Ze względu na wdrożeniowy charakter badań, eksperymenty nad modelami korekty tekstów były prowadzone na danych możliwie zbliżonych do tych, z którymi proponowane modele mogą się spotkać po wdrożeniu w warunkach produkcyjnych. Warunki takie spełniają wewnętrzne korpusy mowy firmy Samsung. Użyte zostały korpusy mowy zawierające polecenia dla wirtualnych asystentów dla trzech europejskich języków: niemieckiego (de-DE), hiszpańskiego (es-ES) i włoskiego (it-IT). Wybór języków był podyktowany celami biznesowymi. Nagrania znajdujące się w tych korpusach pochodziły z dwóch źródeł: część została dokonana w studio w celach rozwoju modeli mowy a część pochodziła z danych zbieranych od użytkowników w trakcie używania asystena

	de-DE	**es-ES**	**fr-FR**
WRR	78.07	90.70	93.51%
Zdania	12242	16905	7180
Wyrazy	37955	55567	28004
Śr. dł. zdania	3.1	3.3	3.9

Tabela 4.2: Statystyki użytych danych wewnętrznych firmy Samsung

głosowego, po uprzednim uzyskaniu ich zgody i anonimizacji danych. Nagrania z tych korpusów zostały zamienione na tekst za pomocą trzech wewnętrznych modeli rozpoznawania mowy firmy Samsung typu end-to-end [41], w ramach procesu ich ewaluacji. Dane dla każdego języka zostały zamienione na tekst za pomocą osobnego modelu. Hipotezy z systemu ASR zostały powiązane z odpowiadającymi im zdaniami referencyjnymi z korpusów mowy, tworząc korpus poprawek. Schemat przedstawiający proces trenowania modeli i ich ewaluacji został przedstawiony w rozdziale 2 na rysunku 2.2.

Tabela 4.2 przedstawia statystyki użytych danych. Najliczniejszy z użytych był podzbiór dla języka hiszpańskiego (16 tysięcy zdań), następnie niemieckiego (12 tysięcy zdań), korpus dla języka francuskiego zawierał jedynie 7 tysięcy zdań. Średnia długość zdań wahała się od 3.1 wyrazu dla języka niemieckiego do 3.9 wyrazu dla francuskiego. Zdania są krótkie, ponieważ najczęściej polecenia skierowane do asystentów głosowych są proste [104]. Różnice w jakości rozpoznawania mowy dla tych języków przez ewaluowane modele ASR są znaczące: od 78% WRR dla niemieckiego do 93.5% WRR dla języka francuskiego. Wynikają one z różnej fazy rozwoju poszczególnych modeli ASR (żaden z nich podczas ewaluacji nie był modelem wdrożonym w środowisku produkcyjnym). Przykładowe dane zostały przedstawione w tabeli A.2 w rozdziale A.

4.3.2. Poleval 2020

W celu poddania opracowywanych modeli korekty zewnętrznej ewalaucji, autor zdecydował się wziąć udział w otwartym wyzwaniu „Post-editing and rescoring of automatic speech recognition results" będącym częścią Poleval 2020. Poleval to odbywająca się cyklicznie kampania ewaluacyjna narzędzi do przetwarzania języka polskiego. W roku 2020 zadanie 1. zatytułowane „Post-editing and rescoring of automatic speech recognition results" stawiało przed uczestnikami zadanie stworzenia modelu poprawy błędów popełnionych przez system rozpoznawania mowy dla języka polskiego [100].

Dane udostępnione przez organizatorów wyzwania powstały z użyciem nagrań z trzech korpusów mowy:

- korpus Clarin-PL studio [89] - zawiera zróżnicowane zdania czytane w studiu przez wielu ludzi. Ten korpus posłużył również do wytrenowania modelu rozpoznawania mowy użytego do generacji danych do wyzwania,

- Polish Parliament Corpus (PPC) [99] [3] – zawierający przemówienia z polskiego parlamentu,
- podzbiór Polish interpreting corpus (PINC) [62] – zawierający przemówienia z Parlamentu Europejskiego w języku polskim (wygłaszane przez polskich europarlamentarzystów po polsku albo tłumaczone na żywo z angielskiego przez tłumaczy pracujących w parlamencie europejskim. Ten korpus był użyty jako zbiór testowy, z poprawnymi zdaniami ukrytymi przed uczestnikami.

Wybrane nagrania z powyższych korpusów zostały przetworzone przez autorów konkursu na tekst za pomocą modelu *tri3b* [67] wytrenowanego na danych z korpusu Clarin-PL studio [89] przy pomocy pakietu Kaldi [111].

Statystyki zbiorów danych użytych do stworzenia rozwiązania zadania z wyzwania Poleval 2020 zostały przedstawione w tabeli 4.3.

	Clarin-PL studio			PPC	PELCRA-PARL	PINC
	Train	Test	Dev	Train	Train	Eval
Zdania	11222	1362	1229	6752	8066	462
WER	9.59	12.08	12.39	45.57	59.95	27.6
WER wyroczni [4]	3.75	4.72	4.93	30.71	-	17.7
Średnia długość zdań	22	21	21	104	12	169
Min. długość zdań	3	6	7	1	2	70
Maks. długość zdań	55	53	49	341	47	435

Tabela 4.3: Statystyki danych z wyzwania Poleval 2020

4.3.3. Open Challenge for Correcting Errors of Speech Recognition Systems

Idea organizacji otwartych wyzwań w celu umożliwienia szerokiej grupie badaczy konfrontacji swoich rozwiązań przy pomocy niezależnych danych i metod ewaluacji stanowiła motywację do stworzenia i opublikowania zbioru danych i ogłoszenia wyzwania dotyczącego korekty błędów ASR. Zgodnie z wiedzą autora, zorganizowane w 2019 wyzwanie „Open Challenge for Correcting Errors of Speech Recognition Systems" (w skrócie OCCESRS)[71][72] było pierwszym tego typu wyzwaniem nie tylko dla języka polskiego, ale w ogóle dla problemu korekty błędów ASR. Stawia ono uczestnikom za cel stworzenie modelu korekty błędów popełnianych przez system rozpoznawania mowy.

Dane udostępnione w ramach wyzwania składają się z par zdań:

- hipoteza ASR (potencjalnie zawierająca błędy),
- zdanie referencyjne (poprawna transkrypcja nagrania).

W celu stworzenia tych danych wykonano następujące kroki:

[3] `http://clip.ipipan.waw.pl/PPC`

1. Dokonano wyboru 9142 zdań z polskiej wersji portalu Wikinews [146]. Wikinews jest siostrzanym projektem Wikipedii, który umożliwia użytkownikom publikację artykułów prasowych z bieżącymi informacjami. W momencie powstawania opisywanego zbioru danych serwis zawierał około 16000 artykułów w języku polskim. Zdania powstały poprzez segmentację wybranych losowo artykułów z serwisu za pomocą reguł zapisanych jako wyrażenia regularne.
2. Organizatorzy wyzwania zlecili nagranie wybranych zdań. Nagrania zostały dokonane w studio przez dwoje rodzimych użytkowników języka polskiego, zajmujących się zawodowo nagrywaniem korpusów mowy.
3. Nagrania zdań zostały zweryfikowane poprzez odsłuchanie ich przez profesjonalnych anotatorów. W przypadku niezgodności zdania źródłowego i nagrania (w wyniki pomyłki osoby czytającej zdania podczas nagrywania) dokonywano transkrypcji nagranej wypowiedzi. W ten sposób otrzymano korpus mowy zawierający nagrania 9142 zdań wraz z ich transkrypcjami.
4. Dokonano automatycznej transkrypcji nagrań z otrzymanego korpusu mowy za pomocą systemu rozpoznawania mowy. Użyto wewnętrznego, eksperymentalnego systemu rozpoznawania mowy firmy Samsung.
5. Zarówno zdania referencyjne jak i hipotezy ASR zostały znormalizowane za pomocą następujących reguł odwrotnej normalizacji:
   a) cyfry i inne słowa niestandardowe zostały zamieniane na ich standardowe (mówione) formy.

      „Początek przemarszu będzie miał miejsce o 10:30 na Targu Rybnym, skąd uczestnicy udadzą się na Targ Drzewny pod pomnik Jana III Sobieskiego."
      ⇒ „Początek przemarszu będzie miał miejsce o dziesiątej trzydzieści na Targu Rybnym skąd uczestnicy udadzą się na Targ Drzewny pod pomnik Jana Trzeciego Sobieskiego."
   b) wszystkie znaki interpunkcyjne z wyjątkiem myślników zostały usunięte:

      „Początek przemarszu będzie miał miejsce o dziesiątej trzydzieści na Targu Rybnym, skąd uczestnicy udadzą się na Targ Drzewny pod pomnik Jana Trzeciego Sobieskiego." ⇒ „Początek przemarszu będzie miał miejsce o dziesiątej trzydzieści na Targu Rybnym skąd uczestnicy udadzą się na Targ Drzewny pod pomnik Jana Trzeciego Sobieskiego"
   c) wszystkie liter zostały zamienione na wielkie:

      „Początek przemarszu będzie miał miejsce o dziesiątej trzydzieści na Targu Rybnym skąd uczestnicy udadzą się na Targ Drzewny pod pomnik Jana Trzeciego Sobieskiego" ⇒ „POCZĄTEK PRZEMARSZU BĘDZIE MIAŁ MIEJSCE O DZIESIĄTEJ TRZYDZIEŚCI NA TARGU RYBNYM SKĄD UCZESTNICY UDADZĄ SIĘ NA TARG DRZEWNY POD POMNIK JANA TRZECIEGO SOBIESKIEGO"

   Zastosowanie powyższych zasad normalizacji ma na celu możliwie jak największą redukcję różnić między hipotezą i zdaniem referencyjnym do różnic wynikających z błędów systemu ASR, tak, żeby zadaniem

	Podzbiór trenujący	Podzbiór testowy
Liczba zdań	8142	1000
średni WER	3.94%	4.01%
Podstawienia	2.8%	-
Delecje	0.8%	-
Insercje	0.9%	-
Średni SER	25%	25%
Średnia długość zdań (wyrazy)	15.40	15.10
Najkrótsze zdanie (wyrazy)	2	3
Najdłuższe zdanie (wyrazy)	100	48

Tabela 4.4: Statystyki zbioru danych z wyzwania OCCESRS

uczestników była jedynie korekta błędów, a nie dodatkowo normalizacja tekstu i przywracanie wielkich liter.

6. Zbiór został podzielony na podzbiór trenujący zawierający 8142 losowo wybrane zdania oraz podzbiór testowy zawierający pozostałe 1000 zdań.

Statystyki dotyczące wielkości podzbiorów, długości zdań oraz ilości i rodzajów błędów można znaleźć w tabeli 4.4. Przykłady zdań z wyzwania pokazano w tabeli A.1. Hipotezy mają formę pojedynczych zdań (1-best hypothesis). Zarówno zdania referencyjne jak i hipotezy są poddane normalizacji zgodnie z opisnaą powyżej procedurą. Przy każdym zdaniu w zbiorze podano odniesienie do artykułu, z któreg pochodzi, zgodnie z wymogami licencji, na której udostępnione są artykuły w portalu Wikinews.

4.4. Eksperymenty

W celu ewaluacji zaproponowanego rozwiązania korekty błędów i wyboru optymalnych metod tagowania przeprowadzono serię eksperymentów z danymi opisanymi w poprzednim podrozdziale. Poniżej znajduje się opis tych eksperymentów wraz z ich wynikami.

4.4.1. Modele korekty błędów dla danych firmy Samsung

Niniejsza sekcja opisuje eksperymenty przeprowadzone na wewnętrznych danych firmy Samsung – dane te najlepiej odzwierciedlają warunki, w których modele korekty błędów są wykorzystywane w praktyce. Po pierwsze, częściowo pochodzą z danych użytkowników, tzn. z nagrań wypowiedzi, które prawdziwi użytkownicy wypowiadają do systemu dialogowego. Po drugie, nagrania te są rozpoznawane przez produkcyjny system ASR.

Modele tagowania

Łącznie, w ramach przeprowadzonych eksperymentów, zostało wytrenowanych 6 modeli tagowania, po 2 dla każdego języka:

- BERT - sieć neuronowa typu BERT [140], pretrenowana na danych dla wybranego języka:
  - Gbert [13] dla niemieckiego,
  - Camembert [90] dla francuskiego,
  - Beto [12] dla hiszpańskiego

  z dodaną pojedynczą warstwą liniową na wyjściu. Cała sieć została dotrenowana (ang. *finetunned*) na danych trenujących. Trenownie trwało 6 epok, po tym czasie wyniki na zbiorze walidacyjnym nie ulegały poprawie,
- tager Flair - tager sekwencji z biblioteki Flair (opis w sekcji 3.1, łączący embeddingi Flair [3], pretrenowaną sieć BERT (jak powyżej), sieć rekurencyjną typu LSTM [50] i warstwę CRF [73].

Wyniki

Tabela 4.5 przedstawia wyniki korekty błędów na danych firmy Samsung. Oba testowane rodzaje modeli (BERT i Flair) pozwoliły istotnie zredukować liczbę błędów: względna redukcja Word Error Rate wahała się między 21% a 25%. Wyniki osiągane przez tager Flair były nieznacznie lepsze w przypadku języków hiszpańskiego i francuskiego, w przypadku języka niemieckiego oba tagery osiągają bardzo zbliżone wyniki z niewielką przewagą modelu BERT. Istotne różnice można zaobserwować między wynikami dla poszczególnych języków. Największe zyski korekta błędów osiąga dla języka niemieckiego, dla którego wyjściowe wyniki ASR są najgorsze (78 WRR). Można to wytłumaczyć tym, że większa liczba błędów oznacza więcej przykładów trenujących i więcej łatwych do poprawienia błędów. Im wyniki ASR są bardziej zbliżone do 100% WRR, tym błędy stają się bardziej subtelne. Część z nich może być niemożliwa do naprawienia bez dostępu do nagrań i kontekstu wypowiedzi nawet przez człowieka a część z nich może być fałszywie ujemna na przykład poprzez błędy w anotacji nagrań, przejawiające się błędnymi zdaniami referencyjnymi. Im lepsze wyniki ASR tym trudniej modelowi korekty poprawiać pozostające błędy. Różnica między wyjściowym WRR dla hiszpańskiego (90.7) i francuskiego (93.5) nie jest tak znacząca, co przekłada się również na różnicę między skutecznością korekty. Zysk dla hiszpańskiego (1.95) jest nieco większy niż dla francuskiego (1.46) stanowi on jednak 21% wszystkich błędów w porównaniu z 22.5% dla francuskiego.

Różnica w czasach trenowania między językami wynika głównie z różnic wielkości zbiorów trenujących, ale wpływ może mieć też wielkość użytych pretrenowanych modeli BERT oraz parametry trenowania. Latencja (czas potrzebny na poprawienie jednego zdania) jest mniejsza dla modeli BERT, ale nawet największa jej wartość dla tagera Flair ($29ms$) jest na tyle mała, że umożliwia użycie modelu w systemie ASR działającym w czasie rzeczywistym, na przykład w systemie dialogowym.

		de-DE	**es-ES**	**fr-FR**
ASR	WRR	78.07	90.70	93.51%
BERT	WRR	83.48	92.65	94.97%
	zysk WRR	5.41	1.95	1.46
	wzgl. red. WER	**24.67%**	20.97%	22.50%
	czas trenowania (godziny)	28	40	10
	latencja (ms)	14	14	13
Flair	WRR	83.40	92.86	95.04
	zysk WRR	5.33	2.16	1.53
	wzgl. red. WER	24.30%	**23.23%**	**23.57%**
	czas trenowania	180	154	67
	latencja (ms)	28	29	18

Tabela 4.5: Wyniki korekty błędów na danych firmy Samsung

Wnioski

Przeprowadzone eksperymenty pokazują, że zaproponowana metoda jest w stanie istotnie poprawić wydajność systemu rozpoznawania mowy przy jednoczesnym zachowaniu akceptowalnych w warunkach przemysłowych opóźnień (latencji) wprowadzanych przez model korekty.

4.4.2. Wyzwanie Poleval 2020

Poniższa sekcja opisuje eksperymenty przeprowadzone w ramach udziału w wyzwaniu „Post-editing and rescoring of automatic speech recognition results", w ramach kampanii Poleval 2020 [100].

Dane

Dane dostarczone przez organizatorów wyzwania zostały opisane w sekcji 4.3.2.

W ramach prac nad rozwiązaniem zadania konkursowego Poleval 2020 stworzone zostały dodatkowe dane trenujące, co było dozwolone przez regulamin. Dodatkowe dane powinny możliwie dobrze odpowiadać danym testowym przygotowanym przez organizatorów – powinny pochodzić z tego samego systemu ASR oraz reprezentować podobną domenę. Dlatego wytrenowano model rozpoznawania mowy z pomocą tej samej konfiguracji Kaldi [67] i za pomocą tych samych danych trenujących [89] co organizatorzy wyzwania. Model ten posłużył do transkrypcji nagrań z korpusu PELCRA-PARL citepelcra-parl, który podobnie do korpusów PPC [99] i PINC [62] użytych przez organizatorów, zawiera przemówienia parlamentarne. Dzięki użyciu dodatkowych danych wielkość zbioru trenującego wzrosła z 11222 do 19288 zdań.

Dane w postaci par hipoteza ASR - zdanie referencyjne posłużyły do wygenerowania referencyjnych tagowań operacjami edycyjnymi, zgodnie z procedurą opisaną w sekcji 4.2.3. Statystyki otrzymanych w ten sposób danych przedstawia tabela 4.6.

Model tagowania

Przy użyciu opisanych powyżej danych wytrenowano model tagowania przy pomocy tagera sekwencji z biblioteki Flair [1] (patrz sekcja 3.1), składającego się z wektorowych reprezentacji słów z kontekstem dla języka polskiego (ang. *contextual word embeddings*) [10], dwukierunkowej sieci rekurencyjnej typu LSTM [50] z końcową warstwą CRF (patrz sekcja 3.1). Wybór powyższej architektury modelu tagowania został podyktowany faktem, że podczas prowadzania prac nad rozwiązaniem osiągała on wyniki *state-of-the-art* w innych zadaniach tagowania (częściami mowy [3] oraz encjami nazwanymi [10][2][3]).

Wyniki

Zgłoszenie przygotowane przy pomocy opisanego powyżej modelu okazało się drugim najlepszym zgłoszeniem w konkursie, osiągając WER 24.7, nieznacznie tylko ustępując pierwszemu rozwiązaniu opracowanemu przez zespół z Uniwersytetu Wrocławskiego (WER 24.31). Warto również nadmienić, że zaproponowane rozwiązanie zmieniło tylko 6% zdań, w porównaniu do zwycięskiego rozwiązania, które zmodyfikowało aż 17.2% zdań ze zbioru testowego, mimo nieznacznej różnicy w osiągniętym WER. Wskazuje to na większą precyzję zaproponowanego rozwiązania. Szczegółowe wyniki są zaprezentowane w tabeli 4.7. Omówienie wyników wszystkich rozwiązań zgłoszonych do wyzwania wraz z ich opisem można znaleźć w [100].

Wnioski

Skuteczność zaproponowanej metody "Otaguj i popraw" została pozytywnie zweryfikowana przy użyciu zewnętrznych danych oraz metod i środowiska ewaluacji. W ramach dalszych badań warto przeprowadzić jeszcze raz eksperymenty na zbiorze Poleval 2020 korzystając z nowszych metod tagowania, między innymi dużych modeli językowych, takich jak BERT [28] czy BART [78].

4.4.3. Wyzwanie OCCESRS

Na potrzeby organizacji wyzwania „Open Challenge for Correcting Errors of Speech Recognition Systems" (OCCESRS) przygotowano 3 bazowe rozwiązania problemu korekty błędów ASR. Wszystkie one korzystają z metody "Otaguj i popraw", ale każde korzysta z innego modelu tagowania.

Dane

Wyjściowe dane zostały opisane w sekcji 4.3.3. Z tych wygenerowane zostały operacje edycyjne. Zbiór operacji ograniczono do 257 najczęściej występujących operacji. Statystyki operacji edycyjnych w zbiorze trenującym zostały przedstawione w tabeli 4.8.

Modele tagowania

Użyte zostały następujące modele tagowania:

- Tager Brilla - opisany w sekcji 3.1,
- HerBERT [96] large[5] - sieć neuronowa typu BERT [140], pretrenowana na polskich danych, z dodaną pojedynczą warstwą liniową na wyjściu. Cała sieć została dotrenowana (ang. *finetunned*) na danych trenujących,
- Tager Flair - tager sekwencji z biblioteki Flair (opis w sekcji 3.1, łączący embeddingi Flair dla języka polskiego [10], pretrenowaną sieć typu transformer (HerBERT large) [96], sieć rekurencyjną typu LSTM [50] i warstwę CRF [73].

Dane wykorzystane do trenowania i ewaluacji zostały opisane w sekcji 4.3.3. Do trenowania i ewaluacji rozwiązań bazowych wykorzystano podzbiór trenujący liczący 8142 zdania. Modele tagowania zostały wytrenowane na podzbiorze losowo wybranych 6513 zdań, pozostałe 814 zdań zostało użytych do optymalizacji działania modeli a 815 do finalnej ewaluacji bazowych rozwiązań.

Wyniki

Otrzymane wyniki przedstawia tabela 4.9. Jak widać, już tager Brilla umożliwia osiągnięcie widocznej poprawy wyników. Chociaż względna redukcja WER na poziomie 3% to wynik dwukrotnie gorszy niż osiągany przez metody neuronowe, to jednak tager Brilla jest od nich znacznie szybszy (12 razy szybszy od modelu BERT i ponad 100 razy szybszy od modelu Flair). Czas potrzebny na „wytrenowanie" tagera Brilla jest nieporównanie mniejszy niż pozostałych testowanych modeli.

Wyniki modeli neuronowych nie są zaskakujące – model „Flair" osiąga lepsze wyniki niż „BERT", opłacając to kosztem wolniejszej inferencji. Można to wytłumaczyć faktem, że oba zawierają pretrenowanego transformera HerBERT a model „Flair" dodaje do niego kilka dodatkowych warstw, które z jednej strony umożliwiają dokładniejsze tagowanie, z drugiej strony dokładają spory narzut obliczeniowy. Z tego powodu model zawierający jedynie HerBERT i liniową warstwę wyjściową będzie lepszym wyborem do zastosowań produkcyjnych, w których znaczenie ma mała latencja. Latencja na poziomie osiąganym przez dwa szybsze modele ($2ms$ i $25ms$) jest bardzo dobrym wynikiem, który stwarza możliwość zastosowania tych modeli w produkcyj-

[5] `https://huggingface.co/allegro/herbert-large-cased`

nych systemach ASR działających w czasie rzeczywistym, takich jak modele używane w asystentach dialogowych, biorąc pod uwagę, że latencja samych modeli ASR waha się w przybliżonym przedziale od $200ms$ do około $3000ms$ [153][38][98]. W zależności od zastosowań i użytego modelu ASR, model „Flair" również może być uznany za spełniający ograniczenia produkcyjne.

4.4.4. Wnioski

Stosunkowo niewielki rozmiar danych OCCESRS tłumaczy gorsze wyniki wyrażone jako względna redukcja WER od tych osiągniętych przez zaproponowaną metodę na pozostałych zbiorach. Interesującym rezultatem byłoby wytrenowanie modelu korekty na połączonych danych z wyzwań Poleval 2020 i OCCESRS. Dałoby to odpowiedź na pytanie, czy trenowanie modelu korekty błędów na danych pochodzących z różnych systemów ASR poprawia wyniki korekty, czy wręcz przeciwnie.

liczność	nazwa operacji
1943	del_suffix_3
2070	del_prefix_2
2098	insert_after_z
2151	replace_się
2152	add_suffix_j
2282	insert_before_i
2353	replace_że
2373	add_suffix_u
2377	replace_do
2486	replace_na
2650	add_suffix_e
2650	replace_z
2744	insert_before_w
2750	replace_nie
2886	add_suffix_m
2889	replace_panie
3106	del_prefix_1
3635	replace_suffix_ą
3862	add_suffix_y
3974	replace_to
4141	replace_i
4247	replace_yyy
4320	add_suffix_ch
4422	replace_jest
4883	insert_after_i
4945	replace_suffix_o
4994	replace_w
6305	join
6450	replace_suffix_ę
7252	del_suffix_2
8856	replace_suffix_y
10283	insert_after_w
11697	replace_suffix_a
14491	replace_suffix_e
14747	del_suffix_1
16553	replace_yy
153865	del
313482	UNSUPPORTED_OPERATION
1707728	None

Tabela 4.6: Najczęściej występujące operacje edycyjne w danych Poleval 2020

Korpus		Clarin		PPC	PINC
Podział	Train	Test	Dev	-	Eval
ASR 1best WER	9.59	12.08	12.39	45.64	27.6
WER wyroczni (na kracie)	3.75	4.72	4.93	30.71	17.7
WER po korekcie błędów	-	10.7	-	-	24.7
Redukcja WER	-	1.38	-	-	2.90
Względna redukcja WER	-	11.42%	-	-	10.50%

Tabela 4.7: Wyniki modelu korekty błędów zgłoszonego do wyzwania Poleval 2020, zadanie 1.

liczność	nazwa operacji
10	add_prefix_o
10	add_suffix_a
10	add_suffix_y
10	replace_ii
11	insert_after_i
11	insert_before_z
11	replace_osiemdziesiątego
12	sreplace_sub_ą_a
13	add_suffix_j
13	del_prefix_2
13	sreplace_sub_k_c
13	split_after_0
14	replace_suffix_o
14	replace_ósmego
16	add_suffix_e
17	insert_after_z
20	replace_dziewiątego
21	del_prefix_1
23	replace_suffix_ą
23	replace_roku
24	del_suffix_4
25	del_suffix_2
30	replace_suffix_ę
30	replace_w
31	del_suffix_5
32	insert_after_w
36	del_suffix_3
43	join_-
46	insert_before_w
50	replace_suffix_a
52	add_suffix_m
59	replace_suffix_e
59	replace_suffix_y
61	add_suffix_go
69	del_suffix_1
72	replace_siódmego
83	join
1143	del
2133	UNSUPPORTED_OPERATION
120347	keep
125517	razem tokenów

Tabela 4.8: Najczęściej występujące operacje edycyjne w danych wyzwania OCCESRS

	ASR	tager Brilla	HerBERT	Flair
WER	0.249	0.241	0.234	**0.229**
WER-relaxed	0.244	0.236	0.229	**0.225**
wzgl. red. WER	–	3.21%	6.02%	**8.03%**
wzgl. red. WER-relaxed	–	3.28%	6.15%	**7.79%**
Micro-avg F1	–	0.940	0.971	**0.972**
Czas trenowania	–	**25s**	$3h$	$54m$
Latencja	–	**2ms**	$25ms$	$230ms$

Tabela 4.9: Wyniki korekty błędów na zbiorze OCCESRS. Wyniki z dopiskiem "relaxed" są otrzymane przez porównanie hipotez z referencyjnym zdaniem po ich normalizacji (zamiana liter na małe, usunięcie znaków interpunkcyjnych, przycinanie spacji na początku i końcu zdania, usuwanie podwójnych spacji)

4.5. Wnioski

W rozdziale zaprezentowano podejście do korekty błędów systemów rozpoznawania mowy "Otaguj i popraw". Zostało ono poddane empirycznej ewaluacji za pomocą eksperymentów prowadzonych przy użyciu różnorodnych danych, pochodzących z różnych modeli ASR, w tym danych zewnętrznych, na których przygotowanie autor nie miał wpływu. Wyniki ewaluacji wskazują, że w zależności od danych i użytych metod tagowania, metoda pozwala zredukować od 8% do 24% błędów, jednocześnie wprowadzając nieznaczne opóźnienie. Założenia przyjęte przy tworzeniu metody, umożliwiające precyzyjną kontrolę nad jej działaniem razem z otrzymanymi wynikami eksperymentów sprawiają, że metoda ta nadaje się do zastosowania w środowisku produkcyjnym.

Rozdział 5

Automatyczna normalizacja tekstu

Normalizacja tekstu jest nieodłączną częścią wielu zadań przetwarzania języka naturalnego. Synteza i rozpoznawanie mowy, tłumaczenie automatyczne, wyszukiwanie i ekstrakcja informacji – wszystkie te zadania wymagają zastosowania normalizacji tekstu na danych wejściowych albo wyjściowych. Normalizacja polega na sprowadzeniu tekstu do pewnej, zdefiniowanej z góry, standardowej formy, bez zmiany jego znaczenia [129]. Dzięki jej przeprowadzeniu na danych wejściowych, następujące po niej w ciągu przetwarzania zadania działają spójnie niezależnie od formy tekstu wejściowego. Z kolei zastosowanie normalizacji na wyjściu potoku NLP sprawia, że wynikowy tekst jest bardziej czytelny dla użytkownika.

Kluczowym pojęciem związanym z problemem normalizacji jest pojęcie słów niestandardowych. Są to słowa, których definicji zazwyczaj nie można znaleźć w słowniku i których pisownia nie przekłada się w bezpośredni sposób na wymowę. Przykład słów niestandardowych to cyfry i wyrażenia je zawierające, skróty. Dokładniejszy opis zagadnienia można znaleźć w pracy [129], która wprowadziła ten termin.

Ze względu na kierunek, w którym przeprowadzana jest normalizacja, możemy odróżnić normalizację i odwrotną normalizację. Normalizacja polega na zastąpieniu wszystkich słów niestandardowych formami standardowymi. Jest stosowana między innymi w przypadku syntezy mowy (TTS), gdzie pożądane jest, aby istniało bezpośrednie mapowanie z liter na fonemy. Normalizacja przykładowego zdania może wyglądać następująco:

> Pociąg IC ”Wielkopolanin” relacji Poznań-Warszawa odjedzie z toru 1. przy peronie II o godz. 16:35.
>
> ⇓
>
> pociąg inter city wielkopolanin relacji poznań warszawa odjedzie z toru pierwszego przy peronie drugim o godzinie szesnastej trzydzieści pięć

W powyższym przykładzie dokonano normalizacji:

- wielkości liter – zamieniono wszystkie litery na małe,
- wyrażeń numerycznych – zamiana liczb porządkowych na odpowiadające im formysłowne,
- wyrażeń czasowych,
- skrótów – skrót ”IC” został rozwinięty do formy ”inter city”,

- znaków interpuncyjnych – usunięto cudzysłowy, łącznik zastąpiono spacją

Transformacja tekstu w odwrotnym kierunku nosi nazwę odwrotnej normalizacji (ang. *Inverse Text Normalization*). Przywraca ona słowa niestandardowe sprowadzając tekst do standardowej formy pisanej. Odwrotna normalizacja jest zwykle stosowana na wyjściu procesu, na przykład jako element przetwarzania końcowego w systemie rozpoznawania mowy, ponieważ tekst zwracany przez model rozpoznawania mowy zazwyczaj jest znormalizowany (dzieje się tak wtedy, kiedy model był trenowany z użyciem znormalizowanych tekstów). Dzięki jej zastosowaniu tekst jest łatwiejszy w zrozumieniu dla człowieka.

> pociąg inter city wielkopolanin relacji poznań warszawa odjedzie z toru pierwszego przy peronie drugim o godzinie szesnastej trzydzieści pięć
>
> ⇓
>
> Pociąg IC "Wielkopolanin" relacji Poznań-Warszawa odjedzie z toru 1. przy peronie II o godzinie 16:35.

W powyższym przykładzie dokonano odwrotnej normalizacji:

- wielkości liter – przywrócono wielkie litery,
- wyrażeń numerycznych – liczebniki porządkowe zostały zamienione na odpowiednie formy pisane cyframi rzymskimi lub arabskimi, w zależności od kontestu,
- wyrażeń czasowych – godzina jest zapisana za pomocą cyfr,
- znaków interpunkcyjnych – przywrócono cudzysłów w nazwie pociągu, dodano łącznik w nazwie relacji pociągu.

Zarówno normalizacja (w kierunku TTS) jak i odwrotna normalizacja (w kierunku ASR) są nietrywialnymi zadaniami. Często nie mają jednego dobrego rozwiązania i wymagają określenia zestawu arbitralnie przyjętych reguł normalizacji. Podczas przeprowadzania normalizacji tracimy pewną część informacji przenoszonych przez tekst, więc proces ten nie jest odwracalny. Zastosowanie odwrotnej normalizacji na uprzednio znormalizowanym tekście zazwyczaj nie przywróci pierwotnego tekstu. Na przykład rozwinięcie skrótu ("godz." ⇒ "godzinie") spowoduje utratę informacji o tym, czy w pierwotnym tekście był użyty skrót. Po przeprowadzeniu odwrotnej normalizacji musimy arbitralnie zdecydować, czy zastąpić wyraz "godzinie" jego skróconą formą, czy pozostawić formę rozwiniętą.

Warto tutaj wspomnieć, że normalizacja może dotyczyć też innych aspektów tekstu. Ciekawą odmianą jest normalizacja diachroniczna – polega ona na przepisaniu tekstów historycznych w taki sposób, żeby były zgodne ze współczesnymi zasadami ortografii i pisowni [56].

Standardowo zasady normalizacji definiuje się za pomocą gramatyk lub wyrażeń regularnych, ewentualnie za pomocą zbioru instrukcji warunkowych zawartych w kodzie źródłowym programu [130]. Popularnym narzędziem, umożliwiającym definiowanie reguł przepisywania jednych ciągów na drugie,

na przykład na potrzeby normalizacji, jest Thrax [116]. Narzędzie to kompiluje napisane przez człowieka wyrażenia regularne i zależne od kontekstu reguły przepisywania do postaci transduktorów skończenie-stanowych (ang. Finite State Transducer (FST)) [61], które mogą być potem wykorzystane do efektywnego przepisywania tekstów.

Normalizacja tekstu opierająca się na stworzonych przez ludzi regułach ma swoje ograniczenia. Po pierwsze, ich stworzenie wymaga specjalistycznej wiedzy i umiejętności, zarówno językowych (gramatyka danego języka) jak i technicznych (znajomość formalizmów takich jak Thrax). Przy rozwoju systemów wielojęzycznych te wymagania stają się trudne do spełnienia. Po drugie, zasady normalizacji niektórych wyrażeń są trudne do wyrażenia za pomocą reguł. Gramatyki nadają się dobrze na przykład do zdefiniowania zasad normalizacji liczebnika – występuje tutaj duża regularność i zasady produkcji niemal nieskończonego zbioru liczb można wyrazić za pomocą skończonej liczby reguł zrozumiałych dla człowieka. Bardziej problematyczna jest odwrotna normalizacja wielkich liter oraz interpunkcji. Niektóre zasady odwrotnej normalizacji interpunkcji czy wielkości liter da się wyrazić za pomocą prostych reguł (na przykład: wstaw przecinek przez wyrazem „że", zamień pierwszą literę w zdaniu na wielką). Pozostałe wymagałyby tworzenia i utrzymywania wielkich zbiorów skomplikowanych reguł i list nazw własnych. Znaczna część przypadków wymaga jednak analizy syntaktycznej i semantycznej zdania, aby zdecydować, gdzie postawić przecinek lub zamienić literę na wielką. Dlatego też w rozwiązywaniu tych problemów najlepiej posłużyć się metodami uczenia maszynowego, które na podstawie wystarczającej liczby poprawnie znormalizowanych danych są w stanie nauczyć się zasad normalizacji i przeprowadzać ją automatycznie, bez potrzeby utrzymywania reguł i gramatyk.

W tym rozdziale zostaną omówione eksperymenty przeprowadzone przez autora, mające na celu stworzenie modeli automatycznej odwrotnej normalizacji tekstu zwracanego przez system rozpoznawania mowy, w zakresie przywracania znaków interpunkcyjnych. Ten aspekt normalizacji, obok przywracania wielkich liter, stwarza najwięcej problemów, kiedy próbuje się je rozwiązać metodami regułowymi, a jednocześnie łatwo jest go sformułować jako problem uczenia maszynowego. Do tego dane trenujące dla tych problemów są łatwe w wygenerowaniu – wymagają jedynie posiadania korpusu tekstowego z docelową normalizacją (z poprawnie użytą interpunkcją). Stworzenie danych wejściowych wymaga jedynie usunięcia znaków interpunkcyjnych, co jest trywialnym zadaniem.

5.1. Przywracanie znaków interpunkcyjnych

Znaki przestankowe pełnią ważną rolę w odbiorze tekstu przez człowieka [95]. Nie tylko ułatwiają czytanie, ale mogą też zmienić jego znaczenie. Badania pokazują, że ludzie preferują teksty z interpunkcją [133]. Również mode-

le NLP mogą lepiej działać, kiedy operują na tekście zawierającym znaki interpunkcyjne [60].

Znaki przestankowe nie są bezpośrednio wymawiane. Można domyśleć się ich istnienia w tekście na podstawie cech prozodycznych wypowiedzi, takich jak intonacja, akcenty czy rytm mowy. Większość systemów rozpoznawania mowy nie rozróżnia cech prozodycznych i jest trenowana na danych pozbawionych znaków interpunkcyjnych, dlatego też tekst przez nie zwracany również ich nie zawiera i muszą one zostać przywrócone na etapie przetwarzania końcowego.

Jak wspomniano we wprowadzeniu do niniejszego rozdziału, problem przywracania interpunkcji najlepiej rozwiązać za pomocą metod uczenia maszynowego. Zaproponowana (patrz rozdział 4) przez autora metoda korekty tekstów "Otaguj i popraw" została zaadaptowana do rozwiązania problemu przywracania znaków interpunkcyjnych i nazwana "Otaguj i przywróć".

Stworzenie modelu przywracającego znaki przestankowe było zadaniem postawionym przez organizatorów zadania 1. kampanii Poleval 2021 [101], zatytułowanego "Punctuation restoration from read text". Celem wyzwania było przygotowanie modelu przywracającego znaki interpunkcyjne w wyjściu systemu ASR, dla tekstu czytanego. Zaproponowane przez autora rozwiązanie "Otaguj i przywróć" posłużyło do stworzenia rozwiązania zadania i zostało opisane w artykule [156]. Poniżej opisane zostało wyzwanie, przedstawiono zaproponowane rozwiązanie oraz osiągnięte wyniki.

5.2. Przegląd istniejących rozwiązań

Sproat i in. w pracy [130] ogłaszają otwarte wyzwanie dotyczące normalizacji słów niestandardowych za pomocą rekurencyjnych sieci neuronowych. Praca ta opisuje wyzwania stojące przed problemem automatycznej normalizacji i proponuje bazowe rozwiązania tego problemu. Pogoda i Walkowiak w pracy [110] opisują metodę automatycznego indukowania zbioru etykiet służących do przywracania interpunkcji, bez użycia predefiniowanych klas znaków interpunkcyjnych. Na tak przygotowanych danych dla języka polskiego i angielskiego przeprowadzają eksperymenty z użyciem kilku modeli, osiągając wyniki F1 dochodzące do 89.6. W pracy [133] Suter i in. zajmują się pełną odwrotną normalizacją, modelując razem problemy przywracania interpunkcji, wielkich liter i odwrotnej normalizacji liczb. Zaproponowany przez nich model to model typu sequence-to-sequence o architekturze Transformer. Ponieważ podejście to nie etykietuje oddzielnych tokenów, ale "tłumaczy" całe zdania, metryką raportowaną przez autorów jest Word Error Rate (WER), a nie precyzja i wrażliwość. Podane wartości redukcji WER dla samych znaków interpunkcyjnych wahają się od 13.02 do 16.49, w zależności od języka.

5.3. Dane

W najprostszej wersji wygenerowanie danych dla problemu przywracania interpunkcji wymaga jedynie posiadania korpusu tekstowego z poprawną interpunkcją. Zdania z tego korpusu stanowią docelowe zdania, które powinien zwracać model przywracania interpunkcji. Aby otrzymać zdania wejściowe, wystarczy zastosować zbiór kilku wyrażeń regularnych usuwających z tekstu znaki interpunkcyjne.

Specyfika zadania postawionego w wyzwaniu "Punctuation restoration from read text" umożliwia użycie dodatkowych danych poza samym tekstem. Jak wspomniano wcześniej, wskazówki prozodyczne takie jak akcent, rytm wypowiedzi i intonacja mogą stanowić istotne informacje podczas przywracania znaków interpunkcyjnych. Dlatego autorzy wyzwania oprócz tekstu udostępnili również pliki dźwiękowe oraz znaczniki czasowe dla każdego wyrazu.

Do przygotowania danych posłużył zbiór 38000 tekstów pochodzących z dwóch źródeł: WikiNews i WikTalks. WikiNews [146] to serwis z wiadomościami pisanymi przez wolontariuszy, opisany w sekcji 4.3.3. WikiTalks to korpus tekstowy powstały poprzez pobranie tekstów ze stron dyskusji (ang. *talk pages*)[1] polskiej Wikipedii. Strony dyskusji zawierają wymiany komentarzy na temat danego artykułu Wikipedii wymieniane przez jego autorów podczas prac nad nim.

Spośród wszystkich 38000 tekstów dokonano selekcji, pozostawiając te, których długość mieściła się między 150 a 300 wyrazami. Teksty z podzbioru WikiTalks musiały zawierać co najmniej jeden znak zapytania. Łącznie otrzymano w ten sposób podzbiór liczący około 10000 zdań.

Wybrane teksty zostały ręcznie poprawione przez rodzimych użytkowników języka polskiego, a następnie nagrane. W nagraniach brało udział 121 osób, o głosach żeńskich i męskich. Łączna długość otrzymanych nagrań wyniosła 39 godzin. Nagrania i źródłowe teksty zostały dopasowane za pomocą procedury *force alignment*. Jest to proces podobny do rozpoznawania mowy, z tym że na jego wejściu algorytm otrzymuje, oprócz nagrania zawierającego mowę, również wypowiadany tekst, a jego zadaniem jest zwrócić znaczniki czasowe dla każdego wyrazu, pozwalające dla każdego wyrazu znaleźć odpowiadający mu fragment mowy. Nagrane i dopasowane teksty zostały podzielone na podzbiory train/test-A/test-B, natomiast reszta tekstów utworzyła podzbiór *rest*. Cały zbiór został przez autorów nazwany *WikiPunct*.

Statystyki danych przygotowanych przez autorów wyzwania zostały przedstawione w tabeli 5.1.

[1] `https://pl.wikipedia.org/wiki/Pomoc:Strona_dyskusji`

	Train	Test-A	Rest
Dokumenty	800	200	-
Wyrazy	165 913	40 842	5 199 670
Maks. długość dokumentu (wyrazy)	313	339	-
Długość dokumentów (zdania) (min/avg/max)	2/14/36	3/14/28	-
!	118	23	8328
?	797	149	24262
:	913	323	42 597
-	2448	621	72 598
,	10 132	2498	331 404
.	10 465	2573	370 538

Tabela 5.1: Statystyki zbioru WikiPunct

5.4. Metoda ”Otaguj i przywróć”

Problem normalizacji można potraktować jako specyficzny rodzaj problemu korekty błędów ASR, opisanego w rozdziale 4. Kiedy potraktujemy tekst bez znaków interpunkcyjnych jako potencjalnie błędną hipotezę do naprawienia, a tekst zawierający znaki interpunkcyjne jako zdanie referencyjne, otrzymamy ten sam problem. Aby dostosować metodę ”Otaguj i popraw” do zadania przywracania interpunkcji, należy zmodyfikować zestaw operacji tak, żeby zawierał jedynie operacje wstawiające brakujące znaki interpunkcyjne i dostosować metodę generowania danych trenujących, tak, żeby generowała tekst niezawierający znaków interpunkcyjnych, ale otagowany operacjami edycyjnymi je przywracającymi. Po dostosowaniu zachowania *korektora* aplikującego operacje edycyjne do tekstu otrzymujemy metodę ”Otaguj i przywróć”. Schemat 5.1 przedstawia przykład wykorzystania metody ”Otaguj i przywróć”. Model ASR dla wypowiedzi użytkownika zwraca hipotezę pozbawioną znaków interpunkcyjnych:

```
powiedział że wiedział
```

Tager przypisuje operacje edycyjne i zwraca otagowane zdanie:

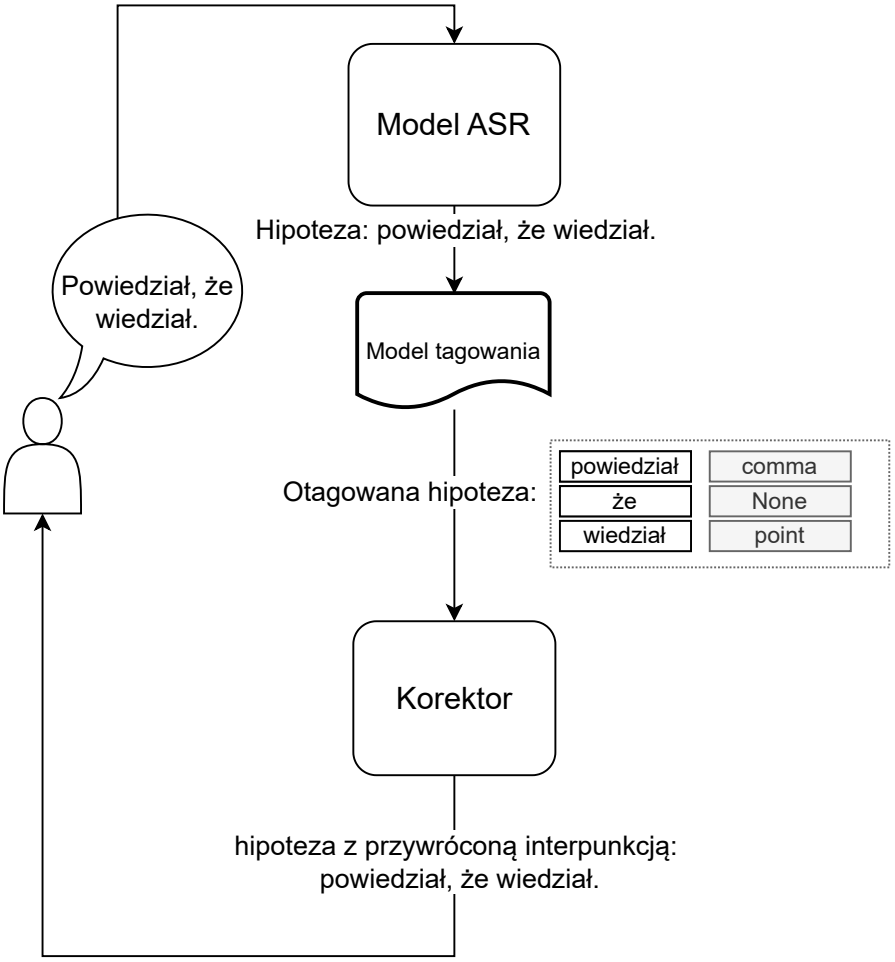
```
Powiedział[comma] że[None] wiedział[point]
```

” Korektor aplikuje operacje do hipotezy, zwracając zdanie z przywróconymi znakami interpunkcyjnymi:

```
powiedział, że wiedział.
```

5.4.1. Przygotowanie danych trenujących

W proponowanym rozwiązaniu, każdemu znakowi interpunkcyjnemu rozważanemu w wyzwaniu odpowiada operacja edycyjna wstawiająca ten znak za wyrazem otagowanym tą operacją edycyjną. Ponieważ w wyzwaniu Poleval podczas oceny rozwiązań brano pod uwagę tylko 7 znaków interpunkcyjnych,



Rysunek 5.1: Metoda "Otaguj i przywróć" użyta do przywrócenia znaków interpunkcyjnych

zbiór operacji ograniczono do siedmiu: po jednej operacji na znak. Wszystkie operacje wraz z przykładami pokazano w tabeli 5.2.

Przygotowanie danych trenujących dla modelu tagowania użytego w metodzie "Otaguj i przywróć" jest wykonywane za pomocą skryptu korzystającego z wyrażeń regularnych, które rozpoznają w tekście referencyjnym znaki interpunkcyjne, usuwają je, a wyrazowi poprzedzającemu usunięty znak przypisują odpowiadającą mu operację edycyjną. Pozostałym wyrazom zostaje przypisana pusta operacja (`None`). Dla przykładowego zdania referencyjnego:

```
Powiedział, że wiedział.
```

przetworzone zdanie zawierające tagi będzie wyglądać następująco:

```
Powiedział[comma] że[None] wiedział[point]
```

Tak przygotowane dane posłużyły do wytrenowania modelu tagowania.

znak	operacja	opis	przykład
	`None`	nie wstawiaj znaku	”kotek” $\xrightarrow{None}$ ”kotek”
.	`point`	wstaw kropkę	”koniec” $\xrightarrow{point}$ ”koniec,”
,	`comma`	wstaw przecinek	”przypuśćmy” $\xrightarrow{comma}$ ”przypuśćmy,”
!	`excl`	wstaw wykrzyknik	”uwaga” $\xrightarrow{excl}$ ”uwaga!”
?	`question`	wstaw znak zapytania	”czyżby” $\xrightarrow{question}$ ”czyżby?”
:	`colon`	wstaw dwukropek	”następujące” $\xrightarrow{colon}$ ”następujące:”
-	`hyphen`	wstaw łącznik	”bielsko” $\xrightarrow{hyphen}$ ”bielsko-”

Tabela 5.2: Operacje edycyjne użyte w metodzie ”Otaguj i przywróć”

5.4.2. Model

Model tagowania wytrenowano bazując na modelu HerBERT [96] – pretrenowanym modelu językowym typu transformer [140] dla języka polskiego. Model został dotrenowany (ang. *finetuned*) na podzbiorze *train* danych udostępnionych przez organizatorów wyzwania.

Niestety, ze względu na udostępnienie przez organizatorów danych trenujących w dwóch lokalizacjach, autor nie użył danych z podzbioru ”rest”, co znacznie ograniczyło liczbę danych użytych do trenowania modeli. Umożliwiło to jednak sprawdzenie wydajności modelu przy użyciu małego zbioru danych.

5.4.3. Wyniki

Wyniki uzyskane przez rozwiązanie podczas wewnętrznej ewaluacji pokazane są w tabeli 5.3. Najlepsze wyniki zaproponowane rozwiązanie osiągnęło dla znaku kropki. Był to też znak najczęściej występujący w danych trenujących (10465 razy), nie dziwi więc wysoka wartość metryki F1 osiągnięta dla tego znaku. Wartość precyzji na poziomie 0.87 oznacza, że w 13% przypadków kropka została postawiona w niewłaściwym miejscu. Wartość recall dla kropki wyniosła 0.94, co oznacza, że kropka nie została przywrócona przez model w 6% miejsc, w których miała być wstawiona. Takie wyniki pozwalają rozważać użycie zaproponowanego modelu w zastosowaniach produkcyjnych. Niestety wyniki dla pozostałych znaków wypadają gorzej. Przecinek, mimo nieznacznie mniej licznych przypadków w zbiorze trenującym, osiąga wartość F1 na poziomie 0.77. Najgorsze wyniki model osiągnął przy przywracaniu wykrzykników, ale ich w zbiorze trenującym było tylko 118.

Przyczyną nie w pełni zadowalających wyników osiągniętych przez model może być niewielka ilość danych trenujących użytych do jego wytrenowania. Mimo to, zaproponowane rozwiązanie osiągnęło średni wynik F1 na danych testowych (nieudostępnionych uczestnikom) na poziomie 78.4 w porównaniu z pozostałymi zgłoszeniami powstałymi przy użyciu ponad 30-krotnie większego zbioru [101], które osiągnęły wartość F1 na poziomie 81.3.

tag	TP	TN	FP	FN	support	prec.	recall	F1
!	1	16140	0	12	13	1.000	0.077	0.143
,	785	14894	273	201	986	0.742	0.796	0.768
-	135	15862	62	94	229	0.685	0.590	0.634
.	1001	14937	150	65	1066	0.870	0.939	0.903
:	58	16037	29	29	87	0.667	0.667	0.667
?	40	16079	11	23	63	0.784	0.635	0.702
None	13362	2198	246	347	13709	0.982	0.975	0.978
Dokładność							0.952	
Średnia ważona						0.793	0.826	0.806

Tabela 5.3: Wyniki wewnętrznej ewaluacji modelu przywracana interpunkcji na podzbiorze test-A

5.5. Wnioski i dalsze prace

W świetle zaprezentowanych wyników osiągniętych na ograniczonym zbiorze danych, wartościowym celem dalszych badań może być zbadanie wpływu wielkości zbioru trenującego na wyniki osiągane przez zaproponowaną metodę przywracania interpunkcji. Warto też zbadać jaki wpływ na wyniki miałaby modyfikacja metody w taki sposób, żeby korzystała ona z informacji o znacznikach czasowych dostępnych dla każdego wyrazu. Znaczniki czasowe niosą informację o początku i końcu trwania każdego wyrazu w wypowiedzi, a co za tym idzie również na temat długości odstępów (pauz) między nimi. Informacja ta może być przydatną wskazówką dla modelu i jej zastosowanie może poprawić jego wyniki.

Rozdział 6

Zastosowania

Opracowane podczas badań metody znalazły liczne zastosowania. Oprócz opisanych w poprzednich częściach pracy zastosowań (wdrożenie w produkcyjnych systemach dialogowych i przygotowanie rozwiązania wyzwań Poleval 2020 [100] i Poleval 2021 [101]), opracowana metoda "Otaguj i przywróć" znalazła też zastosowanie w przeprowadzonych przez autora razem ze współpracownikami pracach nad wpływem błędów ASR na efektywność modeli rozumienia języka naturalnego (NLU). Doświadczenie i umiejętności nabyte podczas prac nad modelami korekty błędów ASR zaowocowały również stworzeniem danych umożliwiających porównywanie wydajności modeli NLU w warunkach zanieczyszczenia danych wejściowych błędami ASR, które zostały opublikowane w formie otwartego wyzwania CAICCAIC.

6.1. Motywacja

Systemy dialogowe komunikujące się z użytkownikiem za pomocą głosu stały się powszechne w XXI wieku. Alexa, Asystent Google, Bixby, Cortana czy Siri – mnogość wirtualnych asystentów na rynku jest tego dowodem. Częścią każdego z takich systemów jest moduł rozpoznawania mowy (dokładny opis architektury systemów dialogowych, patrz rozdział 2). Zwracany przezeń tekst jest przetwarzany przez moduł rozumienia języka naturalnego (NLU). Zgodnie z zasadą separacji problemów (ang. *Separation of concerns*)[1], podczas prac nad modułem NLU w praktyce biznesowej często niejawnie zakłada się, że wejście do modułu NLU (zdanie w języku naturalnym wypowiedziane przez użytkownika) jest wolne od błędów rozpoznawania mowy. Modele NLU są trenowane przy pomocy korpusów zawierających wypowiedzi użytkownika niezawierające błędów rozpoznawania. Dzięki temu rozwój modułów ASR i NLU może przebiegać niezależnie. Takie podejście może jednak prowadzić do pogorszenia wydajności całego systemu dialogowego, w którym moduł NLU otrzymuje na wejściu hipotezy z modułu rozpoznawania mowy zawierające błędy.

[1] Nazwa zaproponowana przez Edsgera W. Dijkstrę [29]. Zasada mówi o podziale problemu na mniejsze, realizowane przez wyspecjalizowane moduły, komunikujące się ze sobą za pomocą interfejsu, który pozwala abstrahować od wewnętrznych szczegółów implementacyjnych innych modułów podczas prac nad danym modułem.

Celem badań nad modelami NLU przeprowadzonych przez autora razem ze współpracownikami było zbadanie wpływu błędów ASR na ich efektywność oraz stworzenie danych i środowiska umożliwiającego porównywanie wydajności tych modeli w warunkach zanieczyszczenia danych wejściowych błędami ASR.

6.2. Dane do rozwoju i ewaluacji modeli NLU odpornych na błędy ASR

Zgodnie z przedstawioną we wprowadzeniu do niniejszej pracy (rozdz. 1) ideą weryfikacji wyników naukowych za pomocą otwartych wyzwań, w celu znalezienia modeli rozumienia języka odpornych na błędy rozpoznawania mowy, w wyniku współpracy między Uniwersytetem Adama Mickiewicza a Samsung Research Poland ogłoszone zostało otwarte wyzwanie zatytułowane "Center for Artificial Intelligence Challenge on Conversational AI Correctness" [70] (w skrócie *CAICCAIC*). Wyzwanie stawia przed uczestnikami problem skonstruowania modelu NLU działającego na zdaniach zaszumionych błędami ASR. Modele są oceniane przy pomocy stworzonego na potrzeby zadania korpusu NLU, zawierającego zdania w języku naturalnym zanieczyszczone błędami modelu ASR.

Autor niniejszej rozprawy w ramach prac nad przygotowaniem wyzwania był odpowiedzialny za:

- przygotowanie za pomocą procedury back-transcription danych opublikowane w ramach wyzwania, w tym dobór i użycie modeli ASR i TTS,
- ewaluację wyników uczestników, w tym definicję i zaimplementowanie metryk ewaluacyjnych oraz analizę wyników

6.2.1. Dane

Dane do wyzwania zostały przygotowane w oparciu o korpus Leyzer [127]. Zawiera on zdania w trzech językach naturalnych: angielskim, polskim i hiszpańskim wraz z ich anotacją semantyczną. Zdania te są skierowane do wirtualnego asystenta, zostały podzielone na 21 domen i 194 zamiary. Przykładowe dane z korpusu Leyzer zostały przedstawione w tabeli A.4.

Tabela 6.1 prezentuje statystyki długości zdań w korpusie Leyzer. Wielkość korpusów dla poszczególnych języków jest zbliżona (20 tysięcy - 22 tysięcy wypowiedzi). Zdania dla języka hiszpańskiego są średnio dłuższe (około 13 wyrazów) niż w przypadku zdań w językach angielskim i polskim (około 9 wyrazów). Rozkład długości zdań w poszczególnych podzbiorach (train, test, valid) jest zbliżony.

Język	Pod-zbiór	Liczba zdań	Długość wypowiedzi				
			Średnia	Odch. std.	Min.	Mediana	Maks.
en-US	test	3344	9.951	4.322	1	9	33
	train	13022	9.345	3.718	1	9	33
	valid	3633	9.281	3.799	1	9	30
es-ES	test	3520	13.214	6.110	1	12	36
	train	15043	13.369	6.022	1	12	39
	valid	3546	13.152	5.948	1	12	39
pl-PL	test	3494	8.927	3.059	1	9	22
	train	12753	8.972	3.028	1	9	26
	valid	3498	9.018	3.053	1	9	23

Tabela 6.1: Statystyki liczności i długości zdań w korpusie Leyzer

Augmentacja danych za pomocą techniki back-transcription

W celu umożliwienia rozwoju i ewaluacji modeli NLU odpornych na błędy ASR autor wraz ze współpracownikami stworzył korpus "CAICCAIC", nazwany od pierwszych liter nazwy wyzwania. Powstał on przez zanieczyszczenie wypowiedzi z korpusu Leyzer błędami rozpoznawania mowy za pomocą procedury back-transcription. Jest to procedura umożliwiająca otrzymanie transkrypcji systemu ASR z samych danych tekstowych. Z danych tych syntezowana jest przy pomocy modelu mowa, która następnie zamieniana jest z powrotem w tekst przy pomocy modelu rozpoznawania mowy. Technika ta była wykorzystywana między innymi do tworzenia modeli normalizacji [107] i korekty tekstu [43]. Jest ona inspirowana procedurą *back-translation* [121] używaną przy tworzeniu modeli tłumaczenia maszynowego[2].

Podczas generowania korpusu do wyzwania, jako modele syntezy mowy użyte zostały modele FastSpeech 2[3][115] dla języka angielskiego, VITS [64] dla języka polskiego i Tacotron 2 [123] dla hiszpańskiego, dwa ostanie pochodzące z biblioteki Coqui [30]. O wyborze modeli TTS zadecydowała ich dostępność (modele opern source) oraz jakość. Dla wszystkich trzech języków syntezowana mowa została przetranskrybowana z powrotem na tekst za pomocą modelu rozpoznawania mowy Whisper[4][114]. Model ten został wybrany, ponieważ jest modelem wielojęzycznym, dzięki czemu ten sam model może być użyty do wszystkich 3 języków z korpusu Leyzer i w momencie powstawania wyzwania dawał najlepsze rezultaty pośród dostępnych modeli open source.

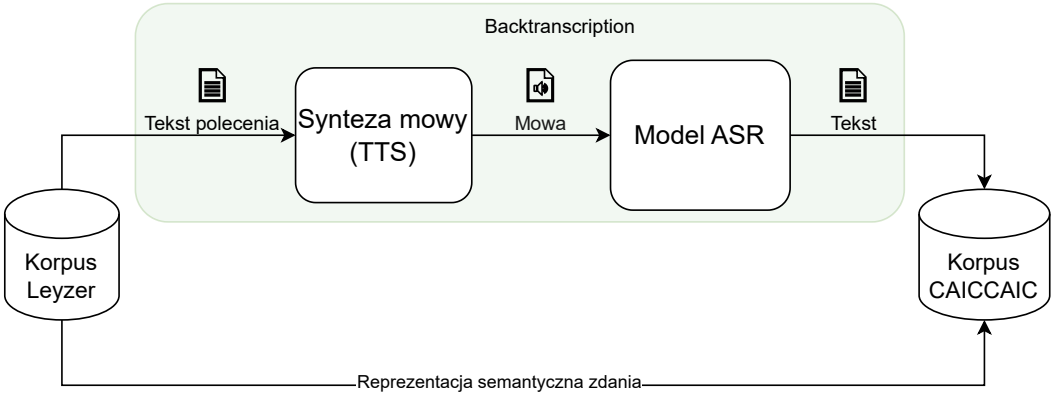
Proces przygotowania danych do korpusu CAICCAIC przedstawiono na schemacie 6.1. Zdania z korpusu Leyzer są zamieniane na mowę za pomocą

[2] Procedura back-translation polega na wzbogaceniu danych trenujących dla problemu tłumaczenia przy użyciu danych jednojęzycznych. Tekst z języka docelowego (A) zostaje przetłumaczony przy pomocy istniejącego modelu na język źródłowy (B) i tak powstały korpus równoległy służy do wzbogacenia danych trenujących model tłumaczenia z języka B na język A.

[3] `https://huggingface.co/facebook/fastspeech2-en-ljspeech`

[4] `https://huggingface.co/openai/whisper-large`

modelu TTS. Powstały w ten sposób sygnał dźwiękowy stanowi wejście moduły rozpoznawania mowy, który zwraca rozpoznany tekst. Hipotezy ASR zostają z powrotem sparowane z reprezentacją semantyczną z korpusu Leyzer (domena, zamiar i wartości slotów) tworząc korpus CAICCAIC.



Rysunek 6.1: Przygotowanie korpusu CAICCAIC

Korpus CAICCAIC został udostępniony uczestnikom i społeczności badaczy[5]. Dane podzielone są na dane wejściowe (w plikach `in.tsv`) i oczekiwane dane wyjściowe (w plikach `expected.tsv`). Dane wejściowe zawierają kod języka i transkrypcję polecenia skierowanego do wirtualnego asystenta. Oczekiwane dane wyjściowe zawierają nazwę domeny, zamiaru oraz słownik z wartościami slotów, jeśli takowe wystąpiły w wypowiedzi. Przykładowe dane zostały zaprezentowane w tabeli A.5.

6.2.2. Rozwiązanie bazowe

Razem ze zbiorem danych autorzy wyzwania zaprezentowali bazowe rozwiązanie problemu. Do klasyfikacji domen, zamiarów i slotów użyte zostały sieci neuronowe, powstałe w oparciu o pretrenowany duży model języka XLM-RoBERTa Base [21]. Jest to wielojęzyczny, neuronowy model języka trenowany na zadaniu MLM (Masked Language Model, patrz opis w sekcji 3.1), przy użyciu oczyszczonych [145] danych Common Crawl [20], zawierających 2.5 TB teksów w ponad 100 językach. Ten pretrenowany model posłużył autorom wyzwania do stworzenia modeli klasyfikacji zamiarów i wypełniania slotów poprzez dotrenowanie sieci na zadaniach klasyfikacji i tagowania przy pomocy danych trenujących CAICCAIC. Ponieważ w korpusie Leyzer zamiar pozwala jednoznacznie zidentyfikować domenę (w całym korpusie nie występują dwa zdania mające przypisaną ten sam zamiar, ale inne domeny), zrezygnowano z trenowania osobnego modelu klasyfikacji domen.

[5] `https://github.com/kubapok/cnlps-caiccaic`

6.2.3. Ewaluacja wyników

Zgłoszenia uczestników były zbierane za pomocą platformy Gonito [39] [6]. Podstawową metryką użytą do ewaluacji wyników była metryka EMA (ang. *Exact Match Accuracy*). Jest ona obliczona jako procent przypadków, dla których odpowiedź zwrócona przez model jest identyczna z oczekiwaną. W przypadku omawianego wyzwania oznacza to, że zarówno domena, zamiar jak i wartości slotów muszą być identyczne z oczekiwanymi, żeby dany przypadek został zaliczony jako poprawny. Dodatkowo obliczone i zaprezentowane uczestnikom zostały również osobne metryki dla zadań klasyfikacji domen (trafność), zamiaru (dokładność) i slotów (WRR na ciągu reprezentującym słownik z wartościami slotów). Metryki zostały policzone za pomocą programu GEval [40], będącego częścią platformy Gonito [39].

6.2.4. Wyniki wyzwania

W wyzwaniu wzięło udział 9 zespołów, nadsyłając łącznie 28 zgłoszeń. W tabeli 6.2 przedstawiono wyniki uzyskane przez model bazowy i przez rozwiązania zgłoszone przez uczestników.[7] Większość zgłoszonych rozwiązań wykorzystała pretrenowane duże modele językowe oparte na architekturze Transformer [140]. Najczęściej używanym modelem był Flan-T5 [17].

Zwycięskie zgłoszenie przedstawione przez Jadczaka i Jaworskiego [54] jako jedyne było oparte na modelu mBART [81]. Jego autorzy zdecydowali się wytrenować jeden łączony model dla domen, zamiarów i slotów, działający jako model sequence-to-sequence, otrzymujący na wejściu zdanie wejściowe w języku naturalnym i zwracający skonkatenowane ciągi reprezentujące domenę, intent i wartości slotów. Brak dominacji zwycięskiego rozwiązania nad bazowym na wszystkich metrykach sugeruje, że połączenie rozwiązania bazowego z wygranym mogłoby przynieść dobre rezultaty. Najprostsze sposób na ich połączenie to zastosowanie rozwiązania bazowego do zdań w języku angielskim i zwycięskiego rozwiązania do poleceń w pozostałych językach. Bardziej wyrafinowane metody, takie jak głosowanie modeli mogłoby zapewnić jeszcze lepsze rezultaty.

Aby dokładniej porównać rozwiązanie bazowe i zwycięskie oraz zrozumieć ich słabe i mocne strony przeprowadzono analizę ich wyników przy pomocy narzędzia Geval [40] i jego funkcji "most worsening feature". Daje ona możliwość analizy wyników na poziomie poszczególnych przypadków i zwraca *cechy* danych wejściowych albo wyjściowych, dla których jeden z porównywanych modeli zwraca wyniki znacząco lepsze od drugiego. Jako *cecha* rozumiana jest obecność danego ciągu w jednej z kolumn danych wejściowych albo wyjściowych. Tablica 6.3 przedstawia 10 najbardziej problematycznych cech, dla których zwycięskie rozwiązanie zwracało gorsze wyniki od bazowego. Pierwsze

[6] `https://gonito.csi.wmi.amu.edu.pl/challenge/cnlps-caiccaic`

[7] Bardziej szczegółowe wyniki zawierające wszystkie zgłoszeni można znaleźć na stronie `https://gonito.csi.wmi.amu.edu.pl/challenge/cnlps-caiccaic/allentries`.

#	zgłoszenie	opis	pl-PL EMA	es-ES EMA	en-US EMA	Slot WRR	Dokł. zamiar	Dokł. domen	EMA
1	8850	mbart-large-50	**0.799**	**0.884**	0.569	**0.872**	0.916	0.963	**0.754**
2	8774	flan-t5-L	0.649	0.787	0.628	0.805	0.922	0.969	0.689
3	8347	*rozw. bazowe*	0.767	0.595	**0.686**	0.752	**0.945**	**0.980**	0.682
4	8812	flan-t5-L+ctx	0.648	0.794	0.548	0.770	0.898	0.955	0.665
5	8687	flan-t5-L	0.550	0.716	0.435	0.738	0.822	0.931	0.569
6	8846	flan-t5	0.495	0.503	0.479	0.692	0.898	0.958	0.493
7	8853	flan-t5	0.516	0.389	0.481	0.626	0.866	0.949	0.461
8	8869	dfd	0.469	0.457	0.411	0.627	0.675	0.959	0.446
8	8856	flan-t5-B	0.463	0.475	0.389	0.624	0.849	0.945	0.443
10	8847	all done	0.344	0.368	0.278	0.451	0.582	0.926	0.331

Tabela 6.2: Wyniki rozwiązań zgłoszonych do wyzwania CAICCAIC

metryka	cecha	liczebność	różnica
dokł. zamiar	in<4>:hundred	357	-0.423
dokł. zamiar	in<4>:eight	224	-0.442
dokł. zamiar	in<4>:six	206	-0.408
dokł. zamiar	in<4>:images	417	-0.290
dokł. zamiar	out:FindImages[..][8]	31	-1.000
dokł. zamiar	in<2>:en-US	3344	-0.090
dokł. zamiar	in<4>:being	55	-0.582
dokł. zamiar	in<4>:small	48	-0.604
dokł. zamiar	out:	2013	-0.103

Tabela 6.3: Cechy sprawiające najwięcej problemów zwycięskiemu modelowi w porównaniu z modelem bazowym, przy porównaniu metryki trafność klasyfikacji zamiaru (ang. *intent*)

3 pozycje sugerują, że model oparty na mBART ma problemu z występującymi w zdaniu wejściowym liczebnikami oraz zamiarem dotyczącym znajdowania obrazów (zamiar `FindImages`). Problematyczne są również zdania w języku angielskim (`n<2>:en-US`), co jest spójne z gorszymi od konkurencji wynikami EMA dla języka angielskiego ukazanymi w tabeli 6.2. Tablica 6.4 ukazuje cechy problematyczne dla modelu bazowego, z którymi lepiej poradził sobie zwycięski model. Większość cech ma związek z zamiarami związanymi z wysyłaniem wiadomości email.

6.2.5. Wnioski i dalsze badania

W celu udoskonalenia ewaluowanych modeli należałoby przeprowadzić bardziej szczegółową analizę błędów. Model powstały z połączenia kilku zaproponowanych modeli, na przykład poprzez głosowanie, mógłby osiągnąć lepsze wyniki. Warto również rozważyć użycie oddzielnych pretrenowanych modeli języka dla różnych języków zamiast modelu wielojęzycznego. W szczególności dla języka

metryka	cecha	liczebność	różnica
EMA	out:subject	707	-0.808
EMA	exp:subject	711	-0.802
EMA	exp:SendEmail[..][9]	766	-0.756
EMA	out:SendEmail[..][10]	771	-0.752
EMA	out:message	835	-0.677
EMA	exp:message	837	-0.671
EMA	in<4>:un	982	-0.609
EMA	exp:to	1020	-0.593
EMA	in<4>:email	748	-0.686
EMA	out:to	885	-0.567

Tabela 6.4: Cechy sprawiające najwięcej problemów modelowi bazowemu w porównaniu ze zwycięskim modelem

angielskiego warto byłoby zastosować rozwiązanie oparte na XLM-Roberta lub Flan-T5.

W ramach dalszych badań nad modelami NLU odpornymi na błędy ASR należałoby przeprowadzić porównanie wydajności zgłoszonych modeli na czystych danych wejściowych, niezawierających błędów i na danych zawierających błędy. Dałoby to odpowiedź na pytania, na ile różnice między modelami wynikają z ich odporności na błędy oraz jaki wpływ na wyniki NLU ma efektywność modelu ASR. Badania nad szczegółową analizą wpływu błędów ASR na modele NLU były celem dalszych badań, przedstawionych w następnym podrozdziale.

6.3. Analiza wpływu błędów ASR na efektywność modeli NLU

Prace nad wyzwaniem CAICCAIC prowadzone przez autora razem ze współpracownikami, opisane w poprzedniej sekcji, zaowocowały stworzeniem potoku przetwarzania umożliwiającego pozyskanie zdań z korpusu NLU zanieczyszczonych przez model rozpoznawania mowy bez konieczności posiadania nagrań poleceń z tego korpusu. Umożliwiły również porównanie wydajności modeli NLU działających na danych zanieczyszczonych w ten sposób. Nie dały jednak odpowiedzi na pytania o wpływ poszczególnych błędów ASR na efektywność modelu NLU. Przeprowadzenie takiej analizy może być bardzo przydatne w kontekście rozwoju systemu dialogowego. Analiza wpływu poszczególnych błędów umożliwia identyfikację błędów rozpoznawania mowy mających największy wpływ na wyniki działania systemu i tym samym priorytetyzację zadań związanych z rozwojem moduł ASR (w tym postprocesora ASR). W tym podrozdziale zostanie opisana zaproponowana [69] przez autora wraz ze współpracownikami metoda umożliwiająca przeprowadzenie takiej analizy.

6.3.1. Metoda

Zaproponowana procedura identyfikacji wpływu błędów ASR na model NLU składa się z trzech części:

- generowanie danych zanieczyszczonych błędami ASR za pomocą procedury back-transcription – część wykonana przez autora rozprawy,
- użycie tak zanieczyszczonych danych do ewaluacji modelu NLU,
- identyfikacja rodzajów błędów ASR mających największy wpływ na wyniki modelu NLU – część wykonana przy użyciu autorskiej metody klasyfikacji błędów za pomocą operacji edycyjnych, będącej częścią metody "Otaguj i popraw" opisanej w rozdziale 4.

Procedura back-transcription

Do stworzenia poleceń z korpusu NLU zanieczyszczonych błędami systemu rozpoznawania mowy służy procedura back-transcription[11]. Polecenia z korpusu są zamieniane na mowę przy pomocy modelu syntezy mowy, a następnie z powrotem na tekst za pomocą modelu rozpoznawania mowy. Dzięki użyciu tej procedury do przeprowadzenia całej analizy nie jest wymagany korpus zawierający nagrania poleceń skierowanych do systemu i można w tym celu użyć dowolnego korpusu NLU.

Identyfikacja błędów ASR mających wpływ na wyniki modelu NLU

W celu identyfikacji błędów wpływających na wyniki modelu NLU, różnice między referencyjną transkrypcją wypowiedzi i transkrypcją powstałą w wyniku procedury back-transcription są zamieniane na ciąg operacji edycyjnych, opisanych w rozdziale 4.2.3. Następnie budowany jest model regresji logistycznej:

$$Y \sim editops(utt, BT(utt))$$

w którym zmienna objaśniana Y opisuje to, czy wynik uzyskany z modelu NLU zastosowanego do wypowiedzi przetworzonej z użyciem procedury back-transcription ulega pogorszeniu w stosunku do wyniku uzyskanego z modelu NLU zastosowanego do wypowiedzi nieprzetworzonej, tj.

$$Y = \begin{cases} 1 & \text{gdy } e(utt) = NLU(utt) \text{ oraz } e(utt) \neq NLU(BT(utt)) \\ 0 & \text{w p.p.} \end{cases}$$

gdzie:

$e(utt)$ oznacza oczekiwane wyjście modelu NLU dla wypowiedzi utt,
$NLU(utt)$ oznacza wynik zwrócony przez model NLU dla wypowiedzi utt,
$BT(utt)$ wypowiedź utt przetworzoną za pomocą procedury back-transcription.

[11] podobnie jak w wyzwaniu CAICCAIC, więcej patrz sekcja 6.2.1

Listing 6.1: Przykład ze zbioru MASSIVE

```
{
    "id": "0",
    "locale": "en-US",
    "partition": "test",
    "scenario": "alarm",
    "intent": "alarm_set",
    "utt": "wake me up at five am this week",
    "annot_utt": "wake me up at [time : five am] [date :
      ↪ this week]",
    "worker_id": "1"
}
```

Zmiennymi objaśniającymi są natomiast operacje edycyjne (*editops*) wygenerowane dla par $(utt, BT(utt))$ przy pomocy metody "Otaguj i popraw" (rozdz. 4.2.3) Tak wytrenowany model regresji dostarcza współczynników regresji odpowiadających poszczególnym operacjom. Współczynniki te umożliwiają jakościowe porównanie wpływu poszczególnych operacji na wyniki NLU.

6.3.2. Eksperymenty

Przeprowadzone eksperymenty miały na celu zaprezentowanie zaproponowanych metod w praktyce. Na poleceniach z podzbioru testowego korpusu NLU MASSIVE [33] przeprowadzono procedurę back-transcription z wykorzystaniem opisanych poniżej modeli TTS i ASR. Tak zaszumione polecenia oraz ich oryginalne wersje stanowiły wejście do wytrenowanego wcześniej modelu NLU. Wyniki uzyskane dla czystych i zaszumionych danych poddano analizie przy pomocy opisanych w poprzednich podrozdziałach metryk i sposobu analizy wpływu błędów ASR na efektywność model NLU. Szczegóły przeprowadzonych eksperymentów i uzyskane wyniki przedstawiono poniżej.

Dane

Do przeprowadzania eksperymentów użyta została anglojęzyczna część korpusu MASSIVE [33]. Jest to wielojęzyczny korpus zawierający polecenia skierowane do asystentów głosowych, powstały przez przetłumaczenie tekstowej części korpusu SLURP [7] z angielskiego na 50 zróżnicowanych typologicznie języków. Korpus zawiera 16521 poleceń, które, podzielonych na 18 domen, 60 zamiarów i zawierających 55 slotów. Przykładowe polecenie z angielskiego podzbioru MASSIVE przedstawiono na listingu 6.1

Modele NLU

Wytrenowano 3 osobne modele NLU służące do klasyfikacji domen, zamiarów i slotów. Modele powstały przez dotrenowanie (ang *finetuning*) pretrenowanego

modelu XLM-RoBERTa [22],[12] na podzbiorze trenującym korpusu MASSIVE (w jego oryginalnej wersji, bez wykonywania procedury back-transcription). Wybór tego modelu był podyktowany możliwością porównania wyników z tymi raportowanymi w artykule opisującym zbiór danych [33] oraz lepszymi wynikami osiąganymi przez modele oparte na XLM-RoBERTa na danych MASSIVE w porównaniu z innymi wielojęzycznymi pretrenowanymi modelami takimi jak mBERT [28] czy DeBERTaV3 [46].

Modele przetwarzania mowy

Procedura back-transcription została przeprowadzona przy użyciu dwóch modeli TTS. Pierwszy z nich, FastSpeech 2 [115], to neuronowy model syntezy mowy wytrenowany z pomocą danych z korpusu LJ Speech [53][13]. Wejściowy tekst jest zamieniany na sekwencję fonemów przy pomocy biblioteki g2pE [108], ta stanowi wejście do enkodera typu transformer [140]. Wyjście ukrytej warstwy enkodera jest dekodowane za pomocą dekodera typu transformer do postaci spektrogramów a te następnie są zamieniane na dźwięk przy pomocy wokodera HiFi-GAN [66]. Drugi użyty model TTS to Tacotron 2 [123][14]. Jest to model wytrenowany przy pomocy tego samego zbioru danych LJ Speech [53] i używa tego samego wokodera HiFi-GAN [66]. W przeciwieństwie do FastSpeech 2, Tacotron 2 jest trenowany bezpośrednio na danych tekstowych, bez użycia biblioteki g2p. Użyta sieć neuronowa ma architekturę enkoder--dekoder, z połączeniem mechanizmem atencji. Zarówno enkoder jak i dekoder zawierają warstwy LSTM [49] i warstwy konwolucyjne(CNN) [35]. Wyjściem sieci są spektrogramy, przekształcane przez wokoder na docelowy sygnał dźwiękowy.

Jako modelu rozpoznawania mowy, który użyty został do zamienienia sygnałów dźwiękowych zwróconych przez modele TTS z powrotem na tekst, użyto modelu Whisper[15][114]. Jest to model korzystający z architektury enkoder-dekoder typu Transofrmer. Został on wytrenowany przy użyciu bardzo dużego w porównaniu z poprzednimi podejściami zbioru danych, zawierającego 680000 godzin nagrań wraz z odpowiadającym im tekstem. [16] Dane zawierają głównie wypowiedzi w języku angielskim, ale 117000 godzin nagrań pochodzi z 96 innych języków. Dzięki temu model ten jest wielojęzyczny. Uzyskanie tak dużej ilości danych było rezultatem przyjęcia podejścia, w którym ilość danych jest ważniejsza od ich jakości. Dlatego jako danych trenujących użyto nagrań powszechnie dostępnych w internecie stosując automatyczne metod odfiltrowywania transkrypcji wątpliwej jakości, na przykład pochodzących z innych systemów ASR. Wyniki raportowane przez twórców modelu sugerują, że skuteczność rozpoznawana przez ten model jest zbliżona do skuteczności ludzkich anotatorów a dodatkowo model ten jest wyjątkowo odporny na za-

[12] Podobnie jak modele opublikowane razem z użytym zbiorem danych MASSIVE [33]. Opis modelu XLM-RoBERTA patrz sekcja 3.1.

[13] `https://huggingface.co/facebook/fastspeech2-en-ljspeech`

[14] `https://huggingface.co/speechbrain/tts-tacotron2-ljspeech`

[15] `https://huggingface.co/openai/whisper-large`

[16] poprzedni największy dostępny publicznie zbiór zawierał "tylko" 31400 godzin [36].

kłócenia w wejściowym sygnale. Dzięki użyciu dużego zbioru zróżnicowanych danych osiąga spójne wyniki na danych pochodzących z różnych źródeł [114].

Wyniki

Wyniki efektywności modeli NLU przed i po zastosowaniu techniki back-transcription, mierzone miarą F1, pokazano w tabeli 6.5. Ten standardowy sposób mierzenia wpływu błędów na efektywność modelu NLU nie daje wglądu w ich przyczyny i wpływ poszczególnych rodzajów błędów.

model NLU	**TTS**	**Metryka**	przed BT	po BT	Δ
domeny	FastSpeech	trafność	0.92	0.88	-0.04
zamiary	FastSpeech	trafność	0.88	0.82	-0.05
sloty	FastSpeech	micro F1	0.80	0.61	-0.19
domeny	Tacotron	trafność	0.92	0.89	-0.03
zamiary	Tacotron	trafność	0.88	0.84	-0.04
sloty	Tacotron	micro F1	0.80	0.61	-0.19

Tabela 6.5: Wyniki ewaluacji modeli NLU przed i po procedurze back-transcription (BT)

Przeprowadzenie klasyfikacji błędów ASR i stworzenie modelu regresji logistycznej przewidującego wpływ poszczególnych błędów ASR na wyniki NLU pozwoliło stworzyć listę najczęściej występujących błędów (tabela 6.6) oraz tych mających najbardziej negatywny wpływ na wynik modelu NLU (tabela 6.7).

W zależności od modelu TTS 14 do 16 błędów mających największy wpływ na pogorszenie odporności nie jest obecna wśród 20 najczęściej występujących błędów. Podczas rozwoju modeli ASR standardową praktyką biznesową jest priorytetyzacja prac zmierzających do naprawy najczęstszych błędów – ich naprawienie ma największy wpływ na metryki używane do oceny modeli ASR. Zamiast tego warto rozważyć użycie rankingów takich jak pokazany w tabeli 6.7 do identyfikacji najważniejszych błędów do naprawienia w modelu ASR – ich naprawienie powinno przynieść lepszy wynik w kontekście ewaluacji całego systemu dialogowego.

6.3.3. Wnioski i dalsze badania

Zaproponowana metoda identyfikacji błędów ASR mających największy wpływ na efektywność modelu NLU może znaleźć zastosowanie w rozwijaniu modeli ASR i NLU współpracujących w ramach jednego systemu dialogowego. Rozwiązaniem wartym zbadania w przyszłości może być zastosowanie do trenowania modelu NLU danych augmentowanych za pomocą metody back-transcription i zbadanie odporności takiego modelu na błędy w porównaniu z modelem trenowanym na czystych danych.

FastSpeech	**Tacotron**
ollie[replace_olly]	mail[add_prefix_e]
mail[add_prefix_e]	a[del]
pm[replace_m.]	ollie[replace_olly]
at[replace_add]	pm[replace_m.]
any[del_suffix_1]	only[sreplace_n_l]
10[replace_ten]	in[del]
and[replace_in]	mails[add_prefix_e]
to[del]	i[del]
a[del]	at[replace_add]
only[sreplace_n_l]	a[add_suffix_n]
9am[del_prefix_1]	4[replace_four]
cnn[replace_n.]	10[replace_ten]
he[add_suffix_y]	at[add_after_six]
at[add_after_six]	and[del]
and[del]	all[replace_olly]
4[replace_four]	oli[replace_olly]
6am[del_prefix_1]	to-do[split_on_first_-]
i'll[replace_olly]	light[add_suffix_s]
his[del_prefix_1]	the[del]
today's[sreplace_y'_y]	today's[sreplace_y'_y]

Tabela 6.6: 20 najczęstszych błędów ASR w zbiorze

FastSpeech	**Tacotron**
pm[replace_m.]	pm[replace_m.]
chants[replace_suffix_ce]	""[add_before_dot]
bowl[sreplace_w_i]	emmy[replace_amy]
cnn[replace_n.]	to-do[split_on_first_-]
inner[replace_inr]	1[replace_one]
9am[del_prefix_1]	paul's[sreplace_u_we]
dollar[add_before_us]	rare[replace_ray]
jeff[add_suffix_rey]	may[replace_email]
reburnet[replace_burnette]	today's[sreplace_y'_y]
at[add_after_six]	a[join_]
lets[replace_let's]	lice[del]
6am[del_prefix_1]	at[add_after_six]
today's[sreplace_y'_y]	and[replace_suffix_y]
max[replace_macs]	enlightening[replace_lighting]
natie[replace_naty]	pondicherry[sreplace_rr_r]
sassy[replace_prefix_c]	barn[del_suffix_1]
pondicherry[sreplace_rr_r]	yuli[add_suffix_a]
ordered[del_suffix_2]	by[del]
mr[add_suffix_.]	will[del]
it[replace_a]	430[replace_thirty]

Tabela 6.7: 20 błędów najbardziej pogarszających wynik NLU

Rozdział 7

Podsumowanie

W pracy zostały zaprezentowane zaproponowane przez autora metody automatycznej korekty błędów i normalizacji wyjścia z systemów rozpoznawania mowy. Przy ich opracowaniu brany był pod uwagę wdrożeniowy kontekst prowadzonych badań, które odbywały się z myślą o zastosowaniu opracowanych modeli w ramach systemu dialogowego. Praca przedstawia ten kontekst i specyfikę rozwoju modeli korekty ASR w środowisku przemysłowym.

Przedstawiona metoda korekty błędów "Otaguj i popraw" znalazła zastosowanie w produkcyjnych systemach dialogowych firmy Samsung. Posłużyła też do przygotowania rozwiązania otwartego wyzwania dotyczącego korekty błędów, w którym zajęła drugie miejsce. Praca zaprezentowała wyniki licznych eksperymentów przeprowadzonych z użyciem tej metody na zróżnicowanych danych. Metoda "Otaguj i przywróć" wzorowana na metodzie "Otaguj i popraw" posłużyła do stworzenia rozwiązania w wyzwaniu dotyczącym przywracania znaków interpunkcyjnych. Zaproponowana metoda korekty błędów znalazła również zastosowanie podczas badań nad wpływem błędów ASR na efektywność modeli NLU, prowadzonych przez autora i współpracowników.

Opisane podejście do problemów korekty i normalizacji wyjścia z systemu rozpoznawania mowy charakteryzuje się możliwością precyzyjnej kontroli nad działaniem metody, a także możliwością łatwej interpretacji jego działania. Możliwość zastosowania z tą metodą dowolnych modeli tagowania sprawia, że można ją dostosować do środowiska, w którym metoda ma być wdrożona, poprzez wybór modeli o dopasowanych do środowiska wymaganiach obliczeniowych. Cechy te stanowią o użyteczności zaproponowanych metod w środowiskach produkcyjnych.

Umiejętności i doświadczenie związane z tematyką błędów ASR zdobyte podczas prac badawczych umożliwiły autorowi stworzenie, wraz ze współautorami, dwóch zbiorów danych, które stały się podstawą organizacji otwartych wyzwań: wyzwania OCCESRS dotyczącego korekty błędów ASR oraz CAICCAIC, dotyczącego konstrukcji modeli NLU odpornych na błędy.

Uzyskane wyniki stanowią punkt wyjścia do kontynuacji badań w zakresie automatycznej normalizacji i korekty błędów ASR i ich wpływu na działanie modeli NLU w kontekście systemów dialogowych. Dalsze badania mogą dotyczyć rozszerzenia eksperymentów na inne języki oraz użycia innych modeli tagowania, uwzględniających rozwój dziedziny, w szczególności nowych, niedostępnych wcześniej dużych modeli językowych.

Ciekawym pomysłem może być również połączenia zbiorów danych pochodzących z wyzwań Poleval 2020, OCCESRS i CAICCAIC w celu wytrenowania modelu korekty błędów i sprawdzenia jak takie połączenie danych wpływa na wyniki osiągane przez ten model na poszczególnych zbiorach. Dane z wyzwania CAICCAIC mogą też posłużyć do sprawdzenia jak na efektywność modeli NLU wpływa zastosowanie modelu korekty błędów na wejściu do moduły NLU. Warte zbadania mogą być też modele łączące problemy korekty błędów i normalizacji.

Metodę "Otaguj i popraw" można zaadaptować do innych zadań NLP, takich jak korekta błędów gramatycznych, łącznie i dzielenie zdań i generowanie podsumowań, co między innymi umożliwiłoby jej bezpośrednie porównanie z podobnymi metodami [86][85], a także rozszerzyłoby zakres jej zastosowań.

Dodatek A

Przykłady danych

id	train-1
hipoteza	DWUDZIESTEGO CZWARTEGO KWIETNIA BIEŻĄCEGO ROKU ROZMAWIALI O WIKIPEDII INTERNECIE WSPÓŁPRACY KLASYFIKOWANIU WIEDZY KSIĄŻKACH I WŁASNOŚCI **LEKTURA LNEJ**
referencja	DWUDZIESTEGO CZWARTEGO KWIETNIA BIEŻĄCEGO ROKU ROZMAWIALI O WIKIPEDII INTERNECIE WSPÓŁPRACY KLASYFIKOWANIU WIEDZY KSIĄŻKACH I WŁASNOŚCI **INTELEKTUALNEJ**
źródło	https://pl.wikinews.org/w/index.php?curid=27343
id	train-2
hipoteza	**EUROPA POWINNA JĄ** TEŻ ŻE SESJE PE W STRASBURGU SĄ DLA NICH UTRUDNIENIEM BO KOMISJA EUROPEJSKA I RADA UE Z KTÓRYMI PE CIĄGLE WSPÓŁPRACUJE MAJĄ SWOJE STAŁE SIEDZIBY W BRUKSELI
referencja	**EUROPOSŁOWIE PRZYPOMINAJĄ** TEŻ ŻE SESJE PE W STRASBURGU SĄ DLA NICH UTRUDNIENIEM BO KOMISJA EUROPEJSKA I RADA UE Z KTÓRYMI PE CIĄGLE WSPÓŁPRACUJE MAJĄ SWOJE STAŁE SIEDZIBY W BRUKSELI
źródło	https://pl.wikinews.org/w/index.php?curid=21290
id	train-3
hipoteza	DZIESIĄTEGO WRZEŚNIA DWA TYSIĄCE ÓSMEGO ROKU **LECH MAM BLADES** OGŁOSIŁ WYNIKI FINANSOWE TRZECIEGO KWARTAŁU WYNOSZĄCE TRZY I **DZIEWIĘĆDZIESIĄTYCH** MILIARDA DOLARÓW STRAT
referencja	DZIESIĄTEGO WRZEŚNIA DWA TYSIĄCE ÓSMEGO ROKU **LEHMAN BROTHERS** OGŁOSIŁ WYNIKI FINANSOWE TRZECIEGO KWARTAŁU WYNOSZĄCE TRZY I **DZIEWIĘĆ DZIESIĄTYCH** MILIARDA DOLARÓW STRAT
źródło	https://pl.wikinews.org/w/index.php?curid=25282
id	train-4
hipoteza	POCHÓD ROZPOCZĄŁ SIĘ NA PLACU SENATORSKIM ALKAMISTA SENAATINTORILLA A **PIERWSZE** W SZEREGU SZŁA SZKOŁA TAŃCA SAMBY SAMBIC TANSSIKOULU
referencja	POCHÓD ROZPOCZĄŁ SIĘ NA PLACU SENATORSKIM ALKAMISTA SENAATINTORILLA A **PIERWSZA** W SZEREGU SZŁA SZKOŁA TAŃCA SAMBY SAMBIC TANSSIKOULU
źródło	https://pl.wikinews.org/w/index.php?curid=30303
id	train-5
hipoteza	DZIESIĄTEGO PAŹDZIERNIKA W KATOWICKIM SPODKU ODBĘDZIE SIĘ DWUDZIESTA DZIEWIĄTA EDYCJA RAWA BLUES FESTIVAL NAJWIĘKSZEJ BLUESOWEJ IMPREZY TYPU INDOOR W EUROPIE
referencja	DZIESIĄTEGO PAŹDZIERNIKA W KATOWICKIM SPODKU ODBĘDZIE SIĘ DWUDZIESTA DZIEWIĄTA EDYCJA RAWA BLUES FESTIVAL NAJWIĘKSZEJ BLUESOWEJ IMPREZY TYPU INDOOR W EUROPIE
źródło	https://pl.wikinews.org/w/index.php?curid=25476
id	train-6
hipoteza	PRZEPROWADZONE W POŁOWIE GRUDNIA DWUSTRONNE ROZMOWY NIE PRZYNIOSŁY REZULTATU
referencja	PRZEPROWADZONE W POŁOWIE GRUDNIA DWUSTRONNE ROZMOWY NIE PRZYNIOSŁY REZULTATU
źródło	https://pl.wikinews.org/w/index.php?curid=5050

Tabela A.1: Przykłady ze zbioru danych "Open Challenge for Correcting Errors of Speech Recognition Systems"

język	de-DE
hipoteza	zeig mir die neueste folge von american gods
referencja	zeig mir die neueste folge von american gods
język	de-DE
hipoteza	aktuelles tv-programm
referencja	aktuelles tv programm
język	de-DE
hipoteza	summer
referencja	zumachen
język	de-DE
hipoteza	sendereinstellung
referencja	sender einstellung
język	de-DE
hipoteza	papa treu
referencja	paw patrol
język	de-DE
hipoteza	saturday 10
referencja	sender 19
język	es-ES
hipoteza	abrir balance en configuración
referencja	abrir balance en configuración
język	es-ES
hipoteza	concierto de the queen
referencja	concierto del queen
język	es-ES
hipoteza	vaya encontrar un problema
referencja	vaya he encontrado un problema
język	es-ES
hipoteza	tv 1
referencja	tve 1
język	es-ES
hipoteza	pon el volumen a 10
referencja	poner volumen a 10
język	es-ES
hipoteza	búscame blay miton me de george ezra en youtube
referencja	búscame blame it on me de george ezra en youtube
język	fr-FR
hipoteza	aller à couleur dans les paramètres
referencja	aller à couleur dans les paramètres
język	fr-FR
hipoteza	éteindre source un
referencja	éteindre source 1
język	fr-FR
hipoteza	ramener la coupe à la maison
referencja	ramenez la coupe à la maison
język	fr-FR
hipoteza	affiche la liste de appareils
referencja	affiche la liste des appareils
język	fr-FR
hipoteza	will c8
referencja	will smith
język	fr-FR
hipoteza	ouvrir hdmi un
referencja	ouvrir hdmi1

Tabela A.2: Przykłady danych dla problemu korekty błędów ASR z korpusów firmy Samsung

korpus	ClarinPL
id	`SES0001_sent013`
hipoteza	spóźniłem się na to spotkanie ale w recepcji maria zostaje mi kartkę że czekała i że następnego dnia wyjeżdżają do brazylii
referencja	spóźniłem się na to spotkanie ale w recepcji maria zostawiła mi kartkę że czekała i że następnego dnia wyjeżdżają do brazylii
korpus	ClarinPL
id	`SES0040_sent030`
hipoteza	jemu więc w przyszłości przypaść miała wielkopolska pomorze gdańskie i okupowana przez czechów małopolska umowa przewidywała jeszcze trzeciego spadkobiercę był nie młodszy brat łokietka kazimierz łęczycki
referencja	jemu więc w przyszłości przypaść miała wielkopolska pomorze gdańskie i okupowana przez czechów małopolska umowa przewidywała jeszcze trzeciego spadkobiercę był nim młodszy brat łokietka kazimierz łęczycki
korpus	ClarinPL
id	`SES0042_rich001`
hipoteza	rodzeństwo podżegacz ziomek synka mango kuźnia wędzonka znaleźne powiedzą my radzenie
referencja	rodzeństwo podżegacz ziomek synka mango kuźnia wędzonka znaleźne powiedzony rdzeń
korpus	ClarinPL
id	`SES0042_sent001`
hipoteza	<unk> profesor niezadowolony że krytykuje się jego metody nauczania zapowiedział odejście z pracy dziekan wydziału ma nadzieję że sprawę uda się jakoś rozwiązać
referencja	profesor nie jest zadowolony że krytykuje się jego metody nauczania zapowiedział odejście z pracy dziekan wydziału ma nadzieję że sprawę uda się jakoś rozwiązać
korpus	PPC
id	`AdamSzejnfeld-20130410-file002`
hipoteza	ale marszałkowska izbą chętnym zasadniczo się pytany o czym nowy kierowano do tronu ministra ale w kończąc tę debatę
referencja	panie marszałku wysoka izbo zdecydowana większość pytań owszem była kierowana do pana ministra ale kończąc tą debatę
korpus	PPC
id	`AdamRybakowicz-20130412-file000`
hipoteza	ani marszałków zdobyli zdrowy panie mniejsze chciałbym zwrócić uwagę na z konta liczną sytuacją dającą się w z obserwować może niż głównie w pociąg regionalnych panice obiecał kontrolę czasu pracy marzeń listów jednak ani słowa o kontroli zapewnienia jakiegokolwiek noclegu podczas przerw w pracy na gminę ręczy sytuację w chce osoba prowadząc odpoczynku tych anglii wytyczone tu stacji docelowych by zasypia fotelu
referencja	panie marszałku wysoka izbo panie ministrze chciałbym zwrócić uwagę na skandaliczną sytuację dającą się zaobserwować u maszynistów głównie w pociągach regionalnych pan minister obiecał kontrolę czasu pracy maszynistów jednak ani słowa o kontroli o zapewnieniu jakiegokolwiek noclegu podczas przerw w pracy nagminną wręcz sytuacją jest gdy osoba prowadząca pociąg po dojechaniu w godzinach wieczornych do stacji docelowej zasypia w fotelu w którym
korpus	PELCRA-PARL
id	`konf_pras_3-0004`
hipoteza	z tej akrę etyka nocy go największy
referencja	cenzury jaką jaka dotyka yy organizacje narodowe na największej
korpus	PELCRA-PARL
id	`konf_pras_3-0005`
hipoteza	obecnych polsce matt formie z łączność owy czyli dna
referencja	obecnej w polsce platformie społecznościowej czyli na
korpus	PELCRA-PARL
id	`konf_pras_3-0006`
hipoteza	w y z boku a a i co związku z tym że kawu jamy w zamierzamy robić
referencja	fejsbuku yy i co związku z tym yy planujemy zamierzamy robić

Tabela A.3: Przykłady danych Poleval 2020 task 1.

domena	Websearch
intent	SearchImagesWithTextOnEngineWithWidthAndWithHeightAndWithCondition
BIO	`o o o o b-img_query i-img_query i-img_query o o o o`
zdanie	pokaż zdjęcia pasujące do tęsknie za tobą i większy niż 800
domena	Weather
intent	SunsetInLocation
BIO	`o o o o b-location`
zdanie	kiedy zachodzi słońce w meksyk
domena	Weather
intent	SunriseInLocation
BIO	`o o o o b-location`
zdanie	czas wschodu słońca dla radomiu
domena	Spotify
intent	PlaySongByArtist
BIO	`o o b-song o b-artist i-artist o o`
zdanie	puść piosenkę irato artysty richard dawson na spotify
domena	Translate
intent	TranslateTextFromLanguageToLanguage
BIO	`o b-text i-text i-text i-text o b-src_lang o b-trg_lang`
zdanie	tłumaczenie do you ship overseas z angielskiego na niemiecki
domena	Contacts
intent	ShowContactWithNameAndWithNumberAndWithType
BIO	`o o o o o o o b-name o o b-phone_number i-phone_number i-phone_number i-phone_number`
zdanie	create a new contact and call it noll with number +1 202 555 0197 as private
domena	Translate
intent	TranslateTextFromLanguageToLanguageWithEngine
BIO	`o o b-text o b-trg_lang o b-src_lang o b-translator`
zdanie	translation of señora to german from spanish with yandex
domena	Phone
intent	CallNumber
BIO	`o b-phone_number i-phone_number i-phone_number i-phone_number`
zdanie	phone 34 678 09 94
domena	Slack
intent	SendPictureToChannel
BIO	`o o o o o o b-channel o o b-caption`
zdanie	WithCaptionpost a picture to slack channel concerned with caption room
domena	Slack
intent	SendPictureWithUrlToChannel
BIO	`o b-picture_url o o o b-channel`
zdanie	post bit.ly/bjvfqqj to slack channel editorial
domena	Spotify
intent	AddSongWithNameToPlaylistWithName
BIO	`o o b-song o o b-playlist`
zdanie	guarda pista oxygen a playlist bittersweet
domena	Instagram
intent	ShowLastNumberPictures
BIO	`o o b-count o o o o o`
zdanie	muestra mis 14 imágenes de instagram más recientes
domena	Airconditioner
intent	TurnOn
BIO	`o b-av_alias i-av_alias`
zdanie	enciende la refrigeración
domena	Wikipedia
intent	SearchQuery
BIO	`o b-query i-query o o`
zdanie	busca willow smith en wiki

Tabela A.4: Przykłady danych z korpusu Leyzer

wejście	**id**	4814
	język	en-US
	podzbiór	train
	zdanie	open for me the politics section of the ny times
wyjście	**domena**	News
	intent	ShowNewsFromSection
	sloty	`{'portal': 'ny times', 'section': 'politics'}`

wejście	**id**	30671
	język	es-ES
	podzbiór	train
	zdanie	manda un correo a goyo con el tema is confirmed as a panelist que dice este año va excelente usando mi gmail
wyjście	**domena**	Email
	intent	SendEmailToAddressWithSubjectAndWithMessage
	sloty	`{'message': 'este año va excelente', 'subject': 'is confirmed as a panelist', 'to': 'goyo'}`

wejście	**id**	4599
	język	en-US
	podzbiór	train
	zdanie	search for my insta pictures taken in spreckels
wyjście	**domena**	Instagram
	intent	ShowPicturesWithLocation
	sloty	`{'location': 'spreckels'}`

wejście	**id**	35475
	język	es-ES
	podzbiór	train
	zdanie	en spotify ponme la pista back street groove
wyjście	**domena**	Spotify
	intent	PlaySong
	sloty	`{'song': 'back street groove'}`

wejście	**id**	22940
	język	pl-PL
	podzbiór	train
	zdanie	wyświetl obrazki wada i węższy niż 800
wyjście	**domena**	Websearch
	intent	SearchImagesWithTextOnEngineWithWidthAndWithHeightAndWithCondition
	sloty	`{'img_query': 'wada'}`

wejście	**id**	12205
	język	en-US
	podzbiór	train
	zdanie	jump to ninth item in contents
wyjście	**domena**	Wikipedia
	intent	GoToElementNumber
	sloty	`{}`

Tabela A.5: Przykłady danych z korpusu CAICCAIC

Bibliografia

[1] Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter i Roland Vollgraf. „FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP". W: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. Minneapolis, Minnesota: Association for Computational Linguistics, czer. 2019, s. 54–59. DOI: 10.18653/v1/N19-4010. URL: https://www.aclweb.org/anthology/N19-4010 (cyt. na str. 27, 48, 58).

[2] Alan Akbik, Tanja Bergmann i Roland Vollgraf. „Pooled Contextualized Embeddings for Named Entity Recognition". W: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, czer. 2019, s. 724–728. DOI: 10.18653/v1/N19-1078. URL: https://aclanthology.org/N19-1078 (cyt. na str. 48, 58).

[3] Alan Akbik, Duncan Blythe i Roland Vollgraf. „Contextual String Embeddings for Sequence Labeling". W: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, sierp. 2018, s. 1638–1649. URL: https://aclanthology.org/C18-1139 (cyt. na str. 26, 27, 48, 56, 58).

[4] Rosana Ardila, Megan Branson, Kelly Davis, Michael Kohler, Josh Meyer, Michael Henretty, Reuben Morais, Lindsay Saunders, Francis Tyers i Gregor Weber. „Common Voice: A Massively-Multilingual Speech Corpus". English. W: *Proceedings of the Twelfth Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, maj 2020, s. 4218–4222. ISBN: 979--10-95546-34-4. URL: https://aclanthology.org/2020.lrec-1.520 (cyt. na str. 40).

[5] Dzmitry Bahdanau, Kyunghyun Cho i Yoshua Bengio. „Neural Machine Translation by Jointly Learning to Align and Translate". W: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Red. Yoshua Bengio i Yann LeCun. 2015. URL: http://arxiv.org/abs/1409.0473 (cyt. na str. 40).

[6] Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck,

Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu Pal, Matt Post i Marcos Zampieri. „Findings of the 2019 Conference on Machine Translation (WMT19)". W: *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*. Florence, Italy: Association for Computational Linguistics, sierp. 2019, s. 1–61. DOI: 10.18653/v1/W19-5301. URL: https://aclanthology.org/W19-5301 (cyt. na str. 8).

[7] Emanuele Bastianelli, Andrea Vanzo, Pawel Swietojanski i Verena Rieser. „SLURP: A Spoken Language Understanding Resource Package". W: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, list. 2020, s. 7252–7262. DOI: 10.18653/v1/2020.emnlp-main.588. URL: https://aclanthology.org/2020.emnlp-main.588 (cyt. na str. 85).

[8] Ian Beaver. „Is AI at human parity yet? A case study on speech recognition". W: *AI Magazine* 43.4 (2022), s. 386–389 (cyt. na str. 8).

[9] Hendrik Blockeel i Joaquin Vanschoren. „Experiment Databases: Towards an Improved Experimental Methodology in Machine Learning". W: *Knowledge Discovery in Databases: PKDD 2007*. Red. Joost N. Kok, Jacek Koronacki, Ramon Lopez de Mantaras, Stan Matwin, Dunja Mladenič i Andrzej Skowron. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, s. 6–17. ISBN: 978-3-540-74976-9 (cyt. na str. 9).

[10] Łukasz Borchmann, Andrzej Gretkowski i Filip Graliński. „Approaching nested named entity recognition with parallel LSTM-CRFs". W: *Proceedings of the PolEval 2018 Workshop*. Red. Maciej Ogrodniczuk i Łukasz Kobyliński. Warszawa: Institute of Computer Science, Polish l of Science, 19 paź. 2018, s. 63–73. URL: http://www.borchmann.pl/wp-content/uploads/2018/10/borchmann-lukasz.pdf. published (cyt. na str. 48, 58, 59).

[11] Eric Brill. „A Simple Rule-Based Part of Speech Tagger". W: *Proceedings of the Third Conference on Applied Natural Language Processing*. ANLC '92. Trento, Italy: Association for Computational Linguistics, 1992, s. 152–155. DOI: 10.3115/974499.974526. URL: https://doi.org/10.3115/974499.974526 (cyt. na str. 23).

[12] José Cañete, Gabriel Chaperon, Rodrigo Fuentes, Jou-Hui Ho, Hojin Kang i Jorge Pérez. „Spanish Pre-Trained BERT Model and Evaluation Data". W: *PML4DC at ICLR 2020*. 2020 (cyt. na str. 27, 56).

[13] Branden Chan, Stefan Schweter i Timo Möller. „German's Next Language Model". W: *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, grud. 2020, s. 6788–6796. DOI: 10.18653/v1/2020.coling-main.598. URL: https://aclanthology.org/2020.coling-main.598 (cyt. na str. 27, 56).

[14] Xuankai Chang, Wangyou Zhang, Yanmin Qian, Jonathan Le Roux i Shinji Watanabe. „End-To-End Multi-Speaker Speech Recognition With Transformer". W: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

2020, s. 6134–6138. DOI: 10.1109/ICASSP40776.2020.9054029 (cyt. na str. 31).

[15] Alebachew Chiche i Betselot Yitagesu. „Part of speech tagging: a systematic review of deep learning and machine learning approaches". W: *Journal of Big Data* 9.1 (sty. 2022), s. 10. ISSN: 2196-1115. DOI: 10.1186/s40537-022-00561-y. URL: https://doi.org/10.1186/s40537-022-00561-y (cyt. na str. 23).

[16] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk i Yoshua Bengio. „Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". W: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, paź. 2014, s. 1724–1734. DOI: 10.3115/v1/D14-1179. URL: https://aclanthology.org/D14-1179 (term. wiz. 04. 08. 2023) (cyt. na str. 40).

[17] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le i Jason Wei. *Scaling Instruction-Finetuned Language Models*. 2022. DOI: 10.48550/arXiv.2210.11416. arXiv: 2210.11416 [cs.LG] (cyt. na str. 81).

[18] Jordan Cohen. „Embedded speech recognition applications in mobile phones: Status, trends, and challenges". W: *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2008, s. 5352–5355. DOI: 10.1109/ICASSP.2008.4518869 (cyt. na str. 21).

[19] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu i Pavel Kuksa. „Natural Language Processing (Almost) from Scratch". W: *Journal of Machine Learning Research* 12.76 (2011), s. 2493–2537. URL: http://jmlr.org/papers/v12/collobert11a.html (cyt. na str. 7).

[20] *Common Crawl - Open Repository of Web Crawl Data*. en. URL: https://commoncrawl.org/ (term. wiz. 19. 09. 2023) (cyt. na str. 80).

[21] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer i Veselin Stoyanov. „Unsupervised Cross-lingual Representation Learning at Scale". W: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, lip. 2020, s. 8440–8451. DOI: 10.18653/v1/2020.acl-main.747. URL: https://aclanthology.org/2020.acl-main.747 (cyt. na str. 28, 80).

[22] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Édouard Grave, Myle Ott, Luke Zettlemoyer i Veselin Stoyanov. „Unsupervised Cross-lingual Representation Learning at Scale". W: *Proceedings of the 58th*

Annual Meeting of the Association for Computational Linguistics. 2020, s. 8440–8451 (cyt. na str. 86).

[23] Alexis Conneau i Guillaume Lample. „Cross-lingual Language Model Pretraining". W: *Advances in Neural Information Processing Systems*. Red. H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox i R. Garnett. T. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/c04c19c2c2474dbf5f7ac4372c5b9af1-Paper.pdf (cyt. na str. 28).

[24] Corinna Cortes, L. D. Jackel i Wan-Ping Chiang. „Limits on Learning Machine Accuracy Imposed by Data Quality". W: *Advances in Neural Information Processing Systems*. T. 7. MIT Press, 1994. URL: https://proceedings.neurips.cc/paper_files/paper/1994/hash/1e056d2b0ebd5c878c550da6ac5d3724-Abstract.html (term. wiz. 20. 07. 2023) (cyt. na str. 50).

[25] Horia Cucu, Andi Buzo, Laurent Besacier i Corneliu Burileanu. „Statistical Error Correction Methods for Domain-Specific ASR Systems". W: *Statistical Language and Speech Processing*. Red. Adrian-Horia Dediu, Carlos Martín-Vide, Ruslan Mitkov i Bianca Truthe. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, s. 83–92. ISBN: 978-3-642-39593-2 (cyt. na str. 37).

[26] K. H. Davis, R. Biddulph i S. Balashek. „Automatic Recognition of Spoken Digits". W: *The Journal of the Acoustical Society of America* 24.6 (1952), s. 637–642. DOI: 10.1121/1.1906946. eprint: https://doi.org/10.1121/1.1906946. URL: https://doi.org/10.1121/1.1906946 (cyt. na str. 7).

[27] Ramon Lopez Cozar Delgado i Masahiro Araki. *Spoken, Multilingual and Multimodal Dialogue Systems: Development and Assessment*. en. John Wiley & Sons, sty. 2007. ISBN: 978-0-470-02156-9 (cyt. na str. 13).

[28] Jacob Devlin, Ming-Wei Chang, Kenton Lee i Kristina Toutanova. „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". W: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, czer. 2019, s. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423 (cyt. na str. 26–28, 48, 58, 86).

[29] Edsger W. Dijkstra. „On the Role of Scientific Thought". W: *Selected Writings on Computing: A personal Perspective*. New York, NY: Springer New York, 1982, s. 60–66. ISBN: 978-1-4612-5695-3. DOI: 10.1007/978-1-4612-5695-3_12. URL: https://doi.org/10.1007/978-1-4612-5695-3_12 (cyt. na str. 77).

[30] Gölge Eren i The Coqui TTS Team. *Coqui TTS*. Wer. 1.4. Sty. 2021. DOI: 10.5281/zenodo.6334862. URL: https://github.com/coqui-ai/TTS (cyt. na str. 79).

[31] Rahhal Errattahi, Asmaa El Hannani i Hassan Ouahmane. „Automatic Speech Recognition Errors Detection and Correction: A Review". W: *Procedia Computer Science* 128 (sty. 2018), s. 32–37. DOI: 10.1016/j.procs.2018.03.005 (cyt. na str. 37).

[32] Wiam Fadel, Toumi Bouchentouf, Pierre-André Buvet i Omar Bourja. „Adapting Off-the-Shelf Speech Recognition Systems for Novel Words". W: *Information* 14.3 (2023). ISSN: 2078-2489. DOI: 10.3390/info14030179. URL: https://www.mdpi.com/2078-2489/14/3/179 (cyt. na str. 35).

[33] Jack FitzGerald, Christopher Hench, Charith Peris, Scott Mackie, Kay Rottmann, Ana Sanchez, Aaron Nash, Liam Urbach, Vishesh Kakarala, Richa Singh, Swetha Ranganath, Laurie Crist, Misha Britan, Wouter Leeuwis, Gokhan Tur i Prem Natarajan. „MASSIVE: A 1M-Example Multilingual Natural Language Understanding Dataset with 51 Typologically-Diverse Languages". W: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Toronto, Canada: Association for Computational Linguistics, lip. 2023, s. 4277–4302. DOI: 10.18653/v1/2023.acl-long.235. URL: https://aclanthology.org/2023.acl-long.235 (cyt. na str. 32, 85, 86).

[34] Robert M. French. „Catastrophic forgetting in connectionist networks". W: *Trends in Cognitive Sciences* 3.4 (1999), s. 128–135. ISSN: 1364-6613. DOI: https://doi.org/10.1016/S1364-6613(99)01294-2. URL: https://www.sciencedirect.com/science/article/pii/S1364661399012942 (cyt. na str. 8, 35).

[35] Kunihiko Fukushima. „Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position". en. W: *Biological Cybernetics* 36.4 (kw. 1980), s. 193–202. ISSN: 0340-1200, 1432-0770. DOI: 10.1007/BF00344251. URL: http://link.springer.com/10.1007/BF00344251 (term. wiz. 25. 09. 2023) (cyt. na str. 86).

[36] Daniel Galvez, Greg Diamos, Juan Torres, Keith Achorn, Juan Cerón, Anjali Gopi, David Kanter, Max Lam, Mark Mazumder i Vijay Janapa Reddi. „The People's Speech: A Large-Scale Diverse English Speech Recognition Dataset for Commercial Usage". W: *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*. Red. J. Vanschoren i S. Yeung. T. 1. Curran, 2021. URL: https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/202cb962ac59075b964b07152d234b70-Paper-round1.pdf (cyt. na str. 86).

[37] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats i Yann N. Dauphin. „Convolutional Sequence to Sequence Learning". W: *Proceedings of the 34th International Conference on Machine Learning*. Red. Doina Precup i Yee Whye Teh. T. 70. Proceedings of Machine Learning Research. PMLR, czer. 2017, s. 1243–1252. URL: https://proceedings.mlr.press/v70/gehring17a.html (cyt. na str. 27).

[38] Abhinav Goyal i Nikesh Garera. „Building Accurate Low Latency ASR for Streaming Voice Search in E-commerce". W: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*. Toronto, Canada: Association for Computational Linguistics, lip. 2023, s. 276–283. DOI: 10.18653/

v1/2023.acl-industry.26. URL: https://aclanthology.org/2023.acl-industry.26 (cyt. na str. 60).

[39] Filip Graliński, Rafał Jaworski, Łukasz Borchmann i Piotr Wierzchoń. „Gonito.net - Open Platform for Research Competition, Cooperation and Reproducibility". W: *Branco, António and Nicoletta Calzolari and Khalid Choukri (eds.), Proceedings of the 4REAL Workshop: Workshop on Research Results Reproducibility and Resources Citation in Science and Technology of Language*. 2016, s. 13–20. URL: http://4real.di.fc.ul.pt/wp-content/uploads/2016/04/4REALWorkshopProceedings.pdf (cyt. na str. 9, 81).

[40] Filip Graliński, Anna Wróblewska, Tomasz Stanisławek, Kamil Grabowski i Tomasz Górecki. „GEval: Tool for Debugging NLP Datasets and Models". W: *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Florence, Italy: Association for Computational Linguistics, sierp. 2019, s. 254–262. DOI: 10.18653/v1/W19-4826. URL: https://www.aclweb.org/anthology/W19-4826 (cyt. na str. 9, 81).

[41] Alex Graves i Navdeep Jaitly. „Towards End-To-End Speech Recognition with Recurrent Neural Networks". W: *Proceedings of the 31st International Conference on Machine Learning*. Red. Eric P. Xing i Tony Jebara. T. 32. Proceedings of Machine Learning Research 2. Bejing, China: PMLR, czer. 2014, s. 1764–1772. URL: https://proceedings.mlr.press/v32/graves14.html (cyt. na str. 52).

[42] Jiatao Gu, Changhan Wang i Junbo Zhao. „Levenshtein Transformer". W: *Advances in Neural Information Processing Systems*. Red. H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox i R. Garnett. T. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/675f9820626f5bc0afb47b57890b466e-Paper.pdf (cyt. na str. 38, 40).

[43] Jinxi Guo, Tara N. Sainath i Ron J. Weiss. „A Spelling Correction Model for End-to-end Speech Recognition". W: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, s. 5651–5655. DOI: 10.1109/ICASSP.2019.8683745. URL: https://arxiv.org/pdf/1902.07178 (cyt. na str. 36, 37, 40, 48, 79).

[44] Nitin Gupta, Shashank Mujumdar, Hima Patel, Satoshi Masuda, Naveen Panwar, Sambaran Bandyopadhyay, Sameep Mehta, Shanmukha Guttula, Shazia Afzal, Ruhi Sharma Mittal i Vitobha Munigala. „Data Quality for Machine Learning Tasks". W: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. KDD '21. Virtual Event, Singapore: Association for Computing Machinery, 2021, s. 4040–4041. ISBN: 9781450383325. DOI: 10.1145/3447548.3470817. URL: https://doi.org/10.1145/3447548.3470817 (cyt. na str. 50).

[45] Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, Shujie Liu, Tie-Yan Liu, Renqian Luo, Arul Menezes, Tao Qin, Frank Seide, Xu Tan, Fei Tian, Lijun Wu, Shu-

angzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang i Ming Zhou. „Achieving Human Parity on Automatic Chinese to English News Translation". W: *CoRR* abs/1803.05567 (2018). arXiv: 1803.05567. URL: http://arxiv.org/abs/1803.05567 (cyt. na str. 8).

[46] Pengcheng He, Jianfeng Gao i Weizhu Chen. *DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing*. 2021. arXiv: 2111.09543 [cs.CL] (cyt. na str. 86).

[47] Yanzhang He, Tara N. Sainath, Rohit Prabhavalkar, Ian McGraw, Raziel Alvarez, Ding Zhao, David Rybach, Anjuli Kannan, Yonghui Wu, Ruoming Pang, Qiao Liang, Deepti Bhatia, Yuan Shangguan, Bo Li, Golan Pundak, Khe Chai Sim, Tom Bagby, Shuo-yiin Chang, Kanishka Rao i Alexander Gruenstein. „Streaming End-to-end Speech Recognition for Mobile Devices". W: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, s. 6381–6385. DOI: 10.1109/ICASSP.2019.8682336 (cyt. na str. 21).

[48] Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath i Brian Kingsbury. „Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups". W: *IEEE Signal Processing Magazine* 29.6 (2012), s. 82–97. DOI: 10.1109/MSP.2012.2205597 (cyt. na str. 7).

[49] Sepp Hochreiter i Jürgen Schmidhuber. „Long Short-Term Memory". W: *Neural Computation* 9.8 (list. 1997). _eprint: https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf, s. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735 (cyt. na str. 86).

[50] Sepp Hochreiter i Jürgen Schmidhuber. „Long Short-term Memory". W: *Neural computation* 9 (grud. 1997), s. 1735–80. DOI: 10.1162/neco.1997.9.8.1735 (cyt. na str. 27, 37, 48, 56, 58, 59).

[51] Marek Hrúz, Pavel Campr, Zdenek Krňoul, Milos Železný, Oya Aran i Pinar Santemiz. „Multi-Modal Dialogue System with Sign Language Capabilities". W: *The Proceedings of the 13th International ACM SIGACCESS Conference on Computers and Accessibility*. ASSETS '11. Dundee, Scotland, UK: Association for Computing Machinery, 2011, s. 265–266. ISBN: 9781450309202. DOI: 10.1145/2049536.2049599. URL: https://doi.org/10.1145/2049536.2049599 (cyt. na str. 13).

[52] Ahmed Imran i Kopparapu Sunil. „Speech recognition for resource deficient languages using frugal speech corpus". W: *2012 IEEE International Conference on Signal Processing, Communication and Computing (ICSPCC 2012)*. 2012, s. 750–755. DOI: 10.1109/ICSPCC.2012.6335664 (cyt. na str. 40).

[53] Keith Ito i Linda Johnson. *The LJ Speech Dataset*. https://keithito.com/LJ-Speech-Dataset/. 2017 (cyt. na str. 86).

[54] Szymon Jadczak i Rafał Jaworski. „Boosting conversational AI correctness by accounting for ASR errors using a sequence to sequence

model". W: *Proceedings of the 18th Conference on Computer Science and Intelligence Systems*. 2023 (cyt. na str. 81).

[55] Abhinav Jain, Hima Patel, Lokesh Nagalapatti, Nitin Gupta, Sameep Mehta, Shanmukha Guttula, Shashank Mujumdar, Shazia Afzal, Ruhi Sharma Mittal i Vitobha Munigala. „Overview and Importance of Data Quality for Machine Learning Tasks". W: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD '20. Virtual Event, CA, USA: Association for Computing Machinery, 2020, s. 3561–3562. ISBN: 9781450379984. DOI: 10.1145/3394486.3406477. URL: https://doi.org/10.1145/3394486.3406477 (cyt. na str. 50).

[56] Krzysztof Jassem, Filip Graliński, Tomasz Obrębski i Piotr Wierzchoń. „Automatic Diachronic Normalization of Polish Texts". en. W: *Investigationes Linguisticae* 37 (2017), s. 17–33. ISSN: 1426-188X. DOI: 10.14746/il.2017.37.2. URL: https://pressto.amu.edu.pl/index.php/il/article/view/13397 (term. wiz. 06. 09. 2023) (cyt. na str. 68).

[57] Andrzej Jeziorski, Filip Sawicki, Oleksandr Solop, Michal Junczyk, Marcin Sikora i Tomasz Ziętkiewicz. „Industrial ASR Troubleshooting Tool". W: *Proceedings of the LREC2020 Industry Track*. Marseille, France: European Language Resources Association (ELRA), maj 2020, s. 10–14. URL: https://www.aclweb.org/anthology/2020.lrec2020industrytrack-1.3 (cyt. na str. 20).

[58] David E. Metzener John W. Ratcliff. *Pattern Matching: The Gestalt Approach*. Lip. 1988. URL: https://www.drdobbs.com/database/pattern-matching-the-gestalt-approach/184407970 (cyt. na str. 44).

[59] Michael Johnston, Srinivas Bangalore, Gunaranjan Vasireddy, Amanda Stent, Patrick Ehlen, Marilyn Walker, Steve Whittaker i Preetam Maloor. „MATCH: An Architecture for Multimodal Dialogue Systems". W: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, lip. 2002, s. 376–383. DOI: 10.3115/1073083.1073146. URL: https://aclanthology.org/P02-1048 (cyt. na str. 13).

[60] Bernard E. M. Jones. „Exploring the role of Punctuation in Parsing Natural Text". W: *CoRR* cmp-lg/9505024 (1995). URL: http://arxiv.org/abs/cmp-lg/9505024 (cyt. na str. 70).

[61] Daniel Jurafsky i James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 3rd draft from Jan 7, 2023. 2023. URL: https://web.stanford.edu/~jurafsky/slp3/ (cyt. na str. 9, 13, 30, 69).

[62] Marta Kajzer-Wietrzny, Adriano Ferraresi, Ivaska Ilmari i Silvia Bernardini. *Mediated discourse at the European Parliament: Empirical investigations*. Zenodo, sierp. 2022. DOI: 10.5281/zenodo.6976930. URL: https://doi.org/10.5281/zenodo.6976930 (cyt. na str. 53, 57).

[63] Dennis F. Kibler i Pat Langley. „The Experimental Study of Machine Learning". W: 1991. URL: https://api.semanticscholar.org/CorpusID:18797039 (cyt. na str. 9).

[64] Jaehyeon Kim, Jungil Kong i Juhee Son. „Conditional Variational Autoencoder with Adversarial Learning for End-to-End Text-to-Speech". W: *Proceedings of the 38th International Conference on Machine Learning*. Red. Marina Meila i Tong Zhang. T. 139. Proceedings of Machine Learning Research. PMLR, lip. 2021, s. 5530–5540. URL: https://proceedings.mlr.press/v139/kim21f.html (cyt. na str. 79).

[65] Tom Kocmi, Rachel Bawden, Ondřej Bojar, Anton Dvorkovich, Christian Federmann, Mark Fishel, Thamme Gowda, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Rebecca Knowles, Philipp Koehn, Christof Monz, Makoto Morishita, Masaaki Nagata, Toshiaki Nakazawa, Michal Novák, Martin Popel i Maja Popović. „Findings of the 2022 Conference on Machine Translation (WMT22)". W: *Proceedings of the Seventh Conference on Machine Translation (WMT)*. Abu Dhabi, United Arab Emirates (Hybrid): Association for Computational Linguistics, grud. 2022, s. 1–45. URL: https://aclanthology.org/2022.wmt-1.1 (cyt. na str. 10).

[66] Jungil Kong, Jaehyeon Kim i Jaekyoung Bae. „HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis". W: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS'20. Vancouver, BC, Canada: Curran Associates Inc., 2020. ISBN: 978-1-71382-954-6 (cyt. na str. 86).

[67] Danijel Koržinek, Krzysztof Marasek, Łukasz Brocki i Krzysztof Wołk. *Polish Read Speech Corpus for Speech Tools and Services*. 2017. arXiv: 1706.00245 [cs.CL] (cyt. na str. 53, 57).

[68] Alex Krizhevsky, Ilya Sutskever i Geoffrey E Hinton. „ImageNet Classification with Deep Convolutional Neural Networks". W: *Advances in Neural Information Processing Systems*. Red. F. Pereira, C. J. Burges, L. Bottou i K. Q. Weinberger. T. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf (cyt. na str. 7).

[69] Marek Kubis, Paweł Skórzewski, Marcin Sowański i Tomasz Ziętkiewicz. „Back Transcription as a Method for Evaluating Robustness of Natural Language Understanding Models to Speech Recognition Errors". pl. Artykuł oczekujący na decyzję o publikacji na konferencji EMNLP 2023 (Conference on Empirical Methods in Natural Language Processing). 2023. URL: https://openreview.net/pdf?id=HkXbOUaL4W (cyt. na str. 83).

[70] Marek Kubis, Paweł Skórzewski, Marcin Sowański i Tomasz Ziętkiewicz. „Center for Artificial Intelligence Challenge on Conversational AI Correctness". pl. W: *Preproceedings of the 18th Conference on Computer Science and Intelligence Systems*. 2023, s. 1313–1318. URL: https://annals-csis.org/proceedings/2023/pliks/6058.pdf (cyt. na str. 10, 78).

[71] Marek Kubis, Zygmunt Vetulani, Mikołaj Wypych i Tomasz Ziętkiewicz. „Open Challenge for Correcting Errors of Speech Recognition Systems". W: *Proceedings of the 9th Language and Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*. Red. Zygmunt Vetulani i Patrick Paroubek. Poznań, Poland: Wydawnictwo Nauka i Innowacje, 17 maj. 2019, s. 219–223. ISBN: 978--83-65988-30-0. URL: https://arxiv.org/abs/2001.03041%20https://gonito.net/gitlist/asr-corrections.git/. published (cyt. na str. 53).

[72] Marek Kubis, Zygmunt Vetulani, Mikołaj Wypych i Tomasz Ziętkiewicz. „Open Challenge for Correcting Errors of Speech Recognition Systems". W: *Human Language Technology. Challenges for Computer Science and Linguistics*. Red. Zygmunt Vetulani, Patrick Paroubek i Marek Kubis. Cham: Springer International Publishing, 2022, s. 322–337. ISBN: 978-3-031-05328-3 (cyt. na str. 10, 37, 51, 53).

[73] John D. Lafferty, Andrew McCallum i Fernando C. N. Pereira. „Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data". W: *Proceedings of the Eighteenth International Conference on Machine Learning*. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, s. 282–289. ISBN: 1-55860--778-1. URL: http://dl.acm.org/citation.cfm?id=645530.655813 (cyt. na str. 25, 56, 59).

[74] Pat Langley. „Machine learning as an experimental science". W: *Machine Learning* 3.1 (1988), s. 5–8 (cyt. na str. 9).

[75] Viet-Bac Le, Sopheap Seng, Laurent Besacier i Brigitte Bigi. „Word/sub-word lattices decomposition and combination for speech recognition". W: *IEEE International conference on Acoustics, Speech and Signal Processing*. Las Vegas, United States, 2008, s. 4321–4324. DOI: 10.1109/ICASSP.2008.4518611. URL: https://hal.science/hal-01392533 (cyt. na str. 50).

[76] Yann LeCun, Yoshua Bengio i Geoffrey Hinton. „Deep learning". W: *Nature* 521.7553 (maj 2015), s. 436–444. ISSN: 1476-4687. DOI: 10.1038/nature14539. URL: https://doi.org/10.1038/nature14539 (cyt. na str. 7).

[77] Yichong Leng, Xu Tan, Linchen Zhu, Jin Xu, Renqian Luo, Linquan Liu, Tao Qin, Xiangyang Li, Edward Lin i Tie-Yan Liu. „FastCorrect: Fast Error Correction with Edit Alignment for Automatic Speech Recognition". W: *Advances in Neural Information Processing Systems*. Red. M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang i J. Wortman Vaughan. T. 34. Curran Associates, Inc., 2021, s. 21708–21719. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/b597460c506e8e35fb0cc1c1905dd3bc-Paper.pdf (cyt. na str. 36, 38, 48).

[78] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov i Luke Zettlemoyer. „BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension". W: *Proceedings of the 58th Annual Meeting of the Association for Computational*

Linguistics. Online: Association for Computational Linguistics, lip. 2020, s. 7871–7880. DOI: 10.18653/v1/2020.acl-main.703. URL: https://aclanthology.org/2020.acl-main.703 (cyt. na str. 58).

[79] Junwei Liao, Yu Shi i Yong Xu. „Automatic Speech Recognition Post-Processing for Readability: Task, Dataset and a Two-Stage Pre-Trained Approach". W: *IEEE Access* 10 (2022), s. 117053–117066. DOI: 10.1109/ACCESS.2022.3219838 (cyt. na str. 36).

[80] Guangya Liu, Shiqi Wang, Jianxing Yu i Jian Yin. „A Survey on Multimodal Dialogue Systems: Recent Advances and New Frontiers". W: *2022 5th International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE)*. 2022, s. 845–853. DOI: 10.1109/AEMCSE55572.2022.00170 (cyt. na str. 13).

[81] Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis i Luke Zettlemoyer. „Multilingual Denoising Pre-training for Neural Machine Translation". W: *Transactions of the Association for Computational Linguistics* 8 (2020), s. 726–742. DOI: 10.1162/tacl_a_00343. URL: https://aclanthology.org/2020.tacl-1.47 (cyt. na str. 81).

[82] Andrej Ljolje, Fernando Pereira i Michael Riley. „Efficient general lattice generation and rescoring". en. W: *6th European Conference on Speech Communication and Technology (Eurospeech 1999)*. ISCA, wrz. 1999, s. 1251–1254. DOI: 10.21437/Eurospeech.1999-293. URL: https://www.isca-speech.org/archive/eurospeech_1999/ljolje99b_eurospeech.html (term. wiz. 20. 07. 2023) (cyt. na str. 50).

[83] Octavio Loyola-González. „Black-Box vs. White-Box: Understanding Their Advantages and Weaknesses From a Practical Point of View". W: *IEEE Access* 7 (2019), s. 154096–154113. DOI: 10.1109/ACCESS.2019.2949286 (cyt. na str. 8).

[84] Rao Ma, Mengjie Qian, Mark J. F. Gales i Kate M. Knill. *Adapting an Unadaptable ASR System*. 2023. arXiv: 2306.01208 [eess.AS] (cyt. na str. 35).

[85] Eric Malmi, Yue Dong, Jonathan Mallinson, Aleksandr Chuklin, Jakub Adamek, Daniil Mirylenka, Felix Stahlberg, Sebastian Krause, Shankar Kumar i Aliaksei Severyn. „Text Generation with Text-Editing Models". W: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Tutorial Abstracts*. Seattle, United States: Association for Computational Linguistics, lip. 2022, s. 1–7. DOI: 10.18653/v1/2022.naacl-tutorials.1. URL: https://aclanthology.org/2022.naacl-tutorials.1 (cyt. na str. 38, 40, 90).

[86] Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka i Aliaksei Severyn. „Encode, Tag, Realize: High-Precision Text Editing". W: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, list. 2019, s. 5054–5065. DOI: 10.18653/v1/D19-1510. URL: https://aclanthology.org/D19-1510 (cyt. na str. 38, 40, 43, 90).

[87] Lidia Mangu, Eric Brill i Andreas Stolcke. „Finding consensus in speech recognition: word error minimization and other applications of confusion networks". W: *Computer Speech & Language* 14.4 (2000), s. 373–400. ISSN: 0885-2308. DOI: https://doi.org/10.1006/csla.2000.0152. URL: https://www.sciencedirect.com/science/article/pii/S0885230800901529 (cyt. na str. 50).

[88] André Mansikkaniemi, Peter Smit i Mikko Kurimo. „Automatic Construction of the Finnish Parliament Speech Corpus ". W: *Proc. Interspeech 2017*. 2017, s. 3762–3766. DOI: 10.21437/Interspeech.2017-1115 (cyt. na str. 40).

[89] Krzysztof Marasek, Danijel Koržinek, Łukasz Brocki i Kamila Jankowska-Lorek. *Clarin-PL Studio Corpus (EMU)*. CLARIN-PL digital repository. 2015. URL: http://hdl.handle.net/11321/236 (cyt. na str. 52, 53, 57).

[90] Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah i Benoît Sagot. „CamemBERT: a Tasty French Language Model". W: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, lip. 2020, s. 7203–7219. URL: https://www.aclweb.org/anthology/2020.acl-main.645 (cyt. na str. 27, 56).

[91] Joshua Maynez, Shashi Narayan, Bernd Bohnet i Ryan McDonald. „On Faithfulness and Factuality in Abstractive Summarization". W: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, lip. 2020, s. 1906–1919. DOI: 10.18653/v1/2020.acl-main.173. URL: https://aclanthology.org/2020.acl-main.173 (cyt. na str. 40).

[92] Michael McCloskey i Neal J. Cohen. „Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem". W: red. Gordon H. Bower. T. 24. Psychology of Learning and Motivation. ISSN: 0079-7421. Academic Press, 1989, s. 109–165. DOI: https://doi.org/10.1016/S0079-7421(08)60536-8. URL: https://www.sciencedirect.com/science/article/pii/S0079742108605368 (cyt. na str. 8, 35).

[93] Zhong Meng, Naoyuki Kanda, Yashesh Gaur, Sarangarajan Parthasarathy, Eric Sun, Liang Lu, Xie Chen, Jinyu Li i Yifan Gong. „Internal language model training for domain-adaptive end-to-end speech recognition". W: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, s. 7338–7342 (cyt. na str. 31).

[94] Tomáš Mikolov, Anoop Deoras, Daniel Povey, Lukáš Burget i Jan Černocký. „Strategies for training large scale neural network language models". W: *2011 IEEE Workshop on Automatic Speech Recognition And Understanding*. 2011, s. 196–201. DOI: 10.1109/ASRU.2011.6163930 (cyt. na str. 7).

[95] Nick Moore. „What's the point? The role of punctuation in realising information structure in written English". W: *Functional Linguistics* 3.1 (maj 2016), s. 6. ISSN: 2196-419X. DOI: 10.1186/s40554-016-0029-x.

URL: https://doi.org/10.1186/s40554-016-0029-x (cyt. na str. 69).

[96] Robert Mroczkowski, Piotr Rybak, Alina Wróblewska i Ireneusz Gawlik. „HerBERT: Efficiently Pretrained Transformer-based Language Model for Polish". W: *Proceedings of the 8th Workshop on Balto-Slavic Natural Language Processing*. Kiyv, Ukraine: Association for Computational Linguistics, kw. 2021, s. 1–10. URL: https://www.aclweb.org/anthology/2021.bsnlp-1.1 (cyt. na str. 27, 59, 74).

[97] Arun Narayanan, Ananya Misra, Khe Chai Sim, Golan Pundak, Anshuman Tripathi, Mohamed Elfeky, Parisa Haghani, Trevor Strohman i Michiel Bacchiani. „Toward Domain-Invariant Speech Recognition via Large Scale Training". W: *2018 IEEE Spoken Language Technology Workshop, SLT 2018, Athens, Greece, December 18-21, 2018*. IEEE, 2018, s. 441–447. DOI: 10.1109/SLT.2018.8639610. URL: https://doi.org/10.1109/SLT.2018.8639610 (cyt. na str. 8).

[98] Thai Son Nguyen, Jan Niehues, Eunah Cho, Thanh-Le Ha, Kevin Kilgour, Markus Muller, Matthias Sperber, Sebastian Stueker i Alex Waibel. *Low Latency ASR for Simultaneous Speech Translation*. 2020. arXiv: 2003.09891 [eess.AS] (cyt. na str. 60).

[99] Maciej Ogrodniczuk. „Polish Parliamentary Corpus". W: *Proceedings of the LREC 2018 Workshop \emphParlaCLARIN: Creating and Using Parliamentary Corpora*. Red. Darja Fišer, Maria Eskevich i Franciska de Jong. event-place: Miyazaki, Japan. Paris, France: European Language Resources Association (ELRA), 2018, s. 15–19. ISBN: 979--10-95546-02-3. URL: http://lrec-conf.org/workshops/lrec2018/W2/summaries/11_W2.html (cyt. na str. 53, 57).

[100] Maciej Ogrodniczuk i Łukasz Kobyliński, red. *Proceedings of the PolEval 2020 Workshop*. Warsaw, Poland: Institute of Computer Science, Polish Academy of Sciences, 2020. ISBN: 978-83-63159-29-0. URL: http://2020.poleval.pl/files/poleval2020.pdf (cyt. na str. 10, 11, 51, 52, 57, 58, 77).

[101] Maciej Ogrodniczuk i Łukasz Kobyliński, red. *Proceedings of the PolEval 2021 Workshop*. Warsaw, Poland: Institute of Computer Science, Polish Academy of Sciences, 2021. ISBN: 978-83-63159-31-3. URL: http://poleval.pl/files/poleval2021.pdf (cyt. na str. 10, 11, 70, 74, 77).

[102] Atul Kr. Ojha, A. Seza Doğruöz, Giovanni Da San Martino, Harish Tayyar Madabushi, Ritesh Kumar i Elisa Sartori, red. *Proceedings of the 17th International Workshop on Semantic Evaluation (SemEval--2023)*. Toronto, Canada: Association for Computational Linguistics, lip. 2023. URL: https://aclanthology.org/2023.semeval-1.0 (cyt. na str. 10).

[103] Sharon L. Oviatt i Philip R. Cohen. „The Contributing Influence of Speech and Interaction on Human Discourse Patterns". W: *Intelligent User Interfaces*. New York, NY, USA: Association for Computing Machinery, 1988, s. 69–83. ISBN: 0201503050. URL: https://doi.org/10.1145/107215.128692 (cyt. na str. 15).

[104] Laubheimer Page i Budiu Raluca. *Intelligent Assistants: Users' Attitudes Toward Alexa, Google Assistant, and Siri*. en. URL: https://www.nngroup.com/articles/voice-assistant-attitudes/ (term. wiz. 23. 09. 2023) (cyt. na str. 52).

[105] Artidoro Pagnoni, Vidhisha Balachandran i Yulia Tsvetkov. „Understanding Factuality in Abstractive Summarization with FRANK: A Benchmark for Factuality Metrics". W: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, czer. 2021, s. 4812–4829. DOI: 10.18653/v1/2021.naacl-main.383. URL: https://aclanthology.org/2021.naacl-main.383 (cyt. na str. 40).

[106] Titouan Parcollet, Rogier van Dalen, Shucong Zhang i Sourav Bhattacharya. *Sumformer: A Linear-Complexity Alternative to Self-Attention for Speech Recognition*. 2023. arXiv: 2307.07421 [cs.CL] (cyt. na str. 8, 35).

[107] Chanjun Park, Jaehyung Seo, Seolhwa Lee, Chanhee Lee, Hyeonseok Moon, Sugyeong Eo i Heuiseok Lim. „BTS: Back TranScription for Speech-to-Text Post-Processor using Text-to-Speech-to-Text". W: *Proceedings of the 8th Workshop on Asian Translation (WAT2021)*. Online: Association for Computational Linguistics, sierp. 2021, s. 106–116. DOI: 10.18653/v1/2021.wat-1.10. URL: https://aclanthology.org/2021.wat-1.10 (cyt. na str. 79).

[108] Kyubyong Park i Jongseok Kim. *g2pE*. https://github.com/Kyubyong/g2p. 2019 (cyt. na str. 86).

[109] Pranamya Patil, Hyungtak Choi, Ranjan Samal, Gurpreet Kaur, Manisha Jhawar, Aniruddha Tammewar i Siddhartha Mukherjee. „Efficient Dialog State Tracking Using Gated- Intent based Slot Operation Prediction for On-device Dialog Systems". W: *Proceedings of the 19th International Conference on Natural Language Processing (ICON)*. New Delhi, India: Association for Computational Linguistics, grud. 2022, s. 67–74. URL: https://aclanthology.org/2022.icon-main.9 (cyt. na str. 15).

[110] Michał Pogoda i Tomasz Walkowiak. „Comprehensive Punctuation Restoration for English and Polish". W: *Findings of the Association for Computational Linguistics: EMNLP 2021*. Punta Cana, Dominican Republic: Association for Computational Linguistics, list. 2021, s. 4610–4619. DOI: 10.18653/v1/2021.findings-emnlp.393. URL: https://aclanthology.org/2021.findings-emnlp.393 (cyt. na str. 70).

[111] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer i Karel Vesely. „The Kaldi Speech Recognition Toolkit". W: *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Catalog No.: CFP11SRW-USB. Hilton Waikoloa Village, Big Island, Hawaii, US: IEEE Signal Processing Society, grud. 2011 (cyt. na str. 53).

[112] Maria Priestley, Fionntán O'donnell i Elena Simperl. „A Survey of Data Quality Requirements That Matter in ML Development Pipelines". W:

J. Data and Information Quality 15.2 (czer. 2023). ISSN: 1936-1955. DOI: 10.1145/3592616. URL: https://doi.org/10.1145/3592616 (cyt. na str. 50).

[113] Adam Przepiórkowski, Mirosław Bańko, Rafał L. Górski i Barbara Lewandowska-Tomaszczyk, red. *Narodowy korpus języka polskiego: praca zbiorowa*. pl. Warszawa: Wydawnictwo Naukowe PWN, 2012. ISBN: 978-83-01-16700-4 (cyt. na str. 24).

[114] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine Mcleavey i Ilya Sutskever. „Robust Speech Recognition via Large-Scale Weak Supervision". W: *Proceedings of the 40th International Conference on Machine Learning*. Red. Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato i Jonathan Scarlett. T. 202. Proceedings of Machine Learning Research. PMLR, lip. 2023, s. 28492–28518. URL: https://proceedings.mlr.press/v202/radford23a.html (cyt. na str. 29, 79, 86, 87).

[115] Yi Ren, Chenxu Hu, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao i Tie-Yan Liu. „FastSpeech 2: Fast and High-Quality End-to-End Text to Speech". W: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL: https://openreview.net/forum?id=piLPYqxtWuA (cyt. na str. 79, 86).

[116] Brian Roark, Richard Sproat, Cyril Allauzen, Michael Riley, Jeffrey Sorensen i Terry Tai. „The OpenGrm open-source finite-state grammar software libraries". W: *Proceedings of the ACL 2012 System Demonstrations*. Jeju Island, Korea: Association for Computational Linguistics, lip. 2012, s. 61–66. URL: https://aclanthology.org/P12-3011 (cyt. na str. 39, 69).

[117] Anna Rogers, Olga Kovaleva i Anna Rumshisky. „A Primer in BERTology: What We Know About How BERT Works". W: *Transactions of the Association for Computational Linguistics* 8 (2020), s. 842–866. DOI: 10.1162/tacl_a_00349. URL: https://aclanthology.org/2020.tacl-1.54 (cyt. na str. 27).

[118] George Saon, Gakuto Kurata, Tom Sercu, Kartik Audhkhasi, Samuel Thomas, Dimitrios Dimitriadis, Xiaodong Cui, Bhuvana Ramabhadran, Michael Picheny, Lynn-Li Lim, Bergul Roomi i Phil Hall. „English Conversational Telephone Speech Recognition by Humans and Machines". W: *Proc. Interspeech 2017*. 2017, s. 132–136. DOI: 10.21437/Interspeech.2017-405. URL: http://dx.doi.org/10.21437/Interspeech.2017-405 (cyt. na str. 8).

[119] D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo i Dan Dennison. „Hidden Technical Debt in Machine Learning Systems". W: *Advances in Neural Information Processing Systems*. Red. C. Cortes, N. Lawrence, D. Lee, M. Sugiyama i R. Garnett. T. 28. Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper_files/paper/2015/file/86df7dcfd896fcaf2674f757a2463eba-Paper.pdf (cyt. na str. 16).

[120] Sailik Sengupta, Jason Krone i Saab Mansour. „On the Robustness of Intent Classification and Slot Labeling in Goal-oriented Dialog Systems to Real-world Noise". W: *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI*. Online: Association for Computational Linguistics, list. 2021, s. 68–79. DOI: 10.18653/v1/2021.nlp4convai-1.7. URL: https://aclanthology.org/2021.nlp4convai-1.7 (cyt. na str. 15).

[121] Rico Sennrich, Barry Haddow i Alexandra Birch. „Improving Neural Machine Translation Models with Monolingual Data". W: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, sierp. 2016, s. 86–96. DOI: 10.18653/v1/P16-1009. URL: https://aclanthology.org/P16-1009 (cyt. na str. 79).

[122] Rico Sennrich, Barry Haddow i Alexandra Birch. „Neural Machine Translation of Rare Words with Subword Units". W: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, sierp. 2016, s. 1715–1725. DOI: 10.18653/v1/P16-1162. URL: https://aclanthology.org/P16-1162 (cyt. na str. 27).

[123] Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerrv-Ryan, Rif A. Saurous, Yannis Agiomvrgiannakis i Yonghui Wu. „Natural TTS Synthesis by Conditioning Wavenet on MEL Spectrogram Predictions". W: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018, s. 4779–4783. DOI: 10.1109/ICASSP.2018.8461368 (cyt. na str. 79, 86).

[124] Maria Shugrina. „Formatting Time-Aligned ASR Transcripts for Readability". W: *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Los Angeles, California: Association for Computational Linguistics, czer. 2010, s. 198–206. URL: https://aclanthology.org/N10-1023 (cyt. na str. 8).

[125] Sukhdeep S. Sodhi, Ellie Ka-In Chio, Ambarish Jash, Santiago Ontañón, Ajit Apte, Ankit Kumar, Ayooluwakunmi Jeje, Dima Kuzmin, Harry Fung, Heng-Tze Cheng, Jon Effrat, Tarush Bali, Nitin Jindal, Pei Cao, Sarvjeet Singh, Senqiang Zhou, Tameen Khan, Amol Wankhede, Moustafa Alzantot, Allen Wu i Tushar Chandra. „Mondegreen: A Post-Processing Solution to Speech Recognition Error Correction for Voice Search Queries". W: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. KDD '21. Virtual Event, Singapore: Association for Computing Machinery, 2021, s. 3569–3575. ISBN: 9781450383325. DOI: 10.1145/3447548.3467156. URL: https://doi.org/10.1145/3447548.3467156 (cyt. na str. 35).

[126] Fábio Souza, Rodrigo Frassetto Nogueira i Roberto de Alencar Lotufo. „Portuguese Named Entity Recognition using BERT-CRF". W: *CoRR* abs/1909.10649 (2019). arXiv: 1909.10649. URL: http://arxiv.org/abs/1909.10649 (cyt. na str. 48).

[127] Marcin Sowański i Artur Janicki. „Leyzer: A Dataset for Multilingual Virtual Assistants". W: *Text, Speech, and Dialogue*. Red. Petr Sojka, Ivan Kopeček, Karel Pala i Aleš Horák. Cham: Springer International Publishing, 2020, s. 477–486. ISBN: 978-3-030-58323-1. DOI: 10.1007/978-3-030-58323-1_51 (cyt. na str. 78).

[128] Constantin Spille, Birger Kollmeier i Bernd T. Meyer. „Comparing human and automatic speech recognition in simple and complex acoustic scenes". W: *Computer Speech & Language* 52 (2018), s. 123–140. ISSN: 0885-2308. DOI: https://doi.org/10.1016/j.csl.2018.04.003. URL: http://www.sciencedirect.com/science/article/pii/S0885230817301705 (cyt. na str. 8).

[129] Richard Sproat, Alan W. Black, Stanley Chen, Shankar Kumar, Mari Ostendorf i Christopher Richards. „Normalization of Non-Standard Words". W: *Comput. Speech Lang.* 15.3 (lip. 2001), s. 287–333. ISSN: 0885-2308. DOI: 10.1006/csla.2001.0169. URL: https://doi.org/10.1006/csla.2001.0169 (cyt. na str. 29, 67).

[130] Richard Sproat i Navdeep Jaitly. *RNN Approaches to Text Normalization: A Challenge*. 2017. arXiv: 1611.00068 [cs.CL] (cyt. na str. 8, 68, 70).

[131] Andreas Stolcke i Jasha Droppo. „Comparing Human and Machine Errors in Conversational Speech Transcription". W: *Proc. Interspeech 2017*. 2017, s. 137–141. DOI: 10.21437/Interspeech.2017-1544 (cyt. na str. 8).

[132] Jana Straková, Milan Straka i Jan Hajic. „Neural Architectures for Nested NER through Linearization". W: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, lip. 2019, s. 5326–5331. DOI: 10.18653/v1/P19-1527. URL: https://aclanthology.org/P19-1527 (cyt. na str. 48).

[133] Benjamin Suter i Josef Novak. „Neural Text Denormalization for Speech Transcripts". W: *Proc. Interspeech 2021*. 2021, s. 981–985. DOI: 10.21437/Interspeech.2021-1814 (cyt. na str. 69, 70).

[134] Ilya Sutskever, Oriol Vinyals i Quoc V Le. „Sequence to Sequence Learning with Neural Networks". W: *Advances in Neural Information Processing Systems*. Red. Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence i K. Q. Weinberger. T. 27. Curran Associates, Inc., 2014. URL: https://proceedings.neurips.cc/paper_files/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf (cyt. na str. 7, 40).

[135] Zheng-Hua Tan i Børge Lindberg. „Speech Recognition on Mobile Devices". W: *Mobile Multimedia Processing: Fundamentals, Methods, and Applications*. Red. Xiaoyi Jiang, Matthew Y. Ma i Chang Wen Chen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, s. 221–237. ISBN: 978-3-642-12349-8. DOI: 10.1007/978-3-642-12349-8_13. URL: https://doi.org/10.1007/978-3-642-12349-8_13 (cyt. na str. 20, 21).

[136] *Text Normalization and Inverse Text Normalization with NVIDIA NeMo*. en-US. Wrz. 2022. URL: https://developer.nvidia.com/

blog/text-normalization-and-inverse-text-normalization-with-nvidia-nemo/ (term. wiz. 29. 09. 2023) (cyt. na str. 36).

[137] Guido van Rossum Tim Peters. *difflib — Helpers for computing deltas*. https://docs.python.org/3/library/difflib.html. Accessed: 2022-03-20. 1998 (cyt. na str. 44).

[138] Jesús Tomás, Josep Àngel Mas i Francisco Casacuberta. „A Quantitative Method for Machine Translation Evaluation". W: *Proceedings of the EACL 2003 Workshop on Evaluation Initiatives in Natural Language Processing: are evaluation methods, metrics and resources reusable?* Columbus, Ohio: Association for Computational Linguistics, kw. 2003, s. 27–34. URL: https://aclanthology.org/W03-2804 (cyt. na str. 30).

[139] Nikos Tsourakis i Manny Rayner. „A Corpus for a Gesture-Controlled Mobile Spoken Dialogue System". W: *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*. Istanbul, Turkey: European Language Resources Association (ELRA), maj 2012, s. 1315–1322. URL: http://www.lrec-conf.org/proceedings/lrec2012/pdf/539_Paper.pdf (cyt. na str. 13).

[140] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser i Illia Polosukhin. „Attention is All you Need". W: *Advances in Neural Information Processing Systems*. Red. I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan i R. Garnett. T. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf (cyt. na str. 23, 26, 27, 40, 47, 56, 59, 74, 81, 86).

[141] Robert A. Wagner i Michael J. Fischer. „The String-to-String Correction Problem". W: *J. ACM* 21.1 (sty. 1974), s. 168–173. ISSN: 0004-5411. DOI: 10.1145/321796.321811. URL: https://doi.org/10.1145/321796.321811 (cyt. na str. 30).

[142] Dong Wang i Xuewei Zhang. *THCHS-30 : A Free Chinese Speech Corpus*. 2015. arXiv: 1512.01882 [cs.CL] (cyt. na str. 40).

[143] Ye-Yi Wang, A. Acero i C. Chelba. „Is word error rate a good indicator for spoken language understanding accuracy". W: *2003 IEEE Workshop on Automatic Speech Recognition and Understanding (IEEE Cat. No.03EX721)*. 2003, s. 577–582. DOI: 10.1109/ASRU.2003.1318504 (cyt. na str. 30).

[144] Henry Weld, Xiaoqi Huang, Siqu Long, Josiah Poon i Soyeon Caren Han. „A Survey of Joint Intent Detection and Slot Filling Models in Natural Language Understanding". W: *ACM Comput. Surv.* 55.8 (grud. 2022). ISSN: 0360-0300. DOI: 10.1145/3547138. URL: https://doi.org/10.1145/3547138 (cyt. na str. 13).

[145] Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin i Edouard Grave. „CCNet: Extracting High Quality Monolingual Datasets from Web Crawl Data". English. W: *Proceedings of the Twelfth Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, maj 2020, s. 4003–4012. ISBN: 979-10-95546-

-34-4. URL: https://aclanthology.org/2020.lrec-1.494 (cyt. na str. 80).

[146] Wikimedia Foundation. *Wikinews*. https://pl.wikinews.org. Access date: 2019-03-01. 2019 (cyt. na str. 54, 71).

[147] Wikipedia contributors. *ENIAC — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=ENIAC&oldid=1165145027. [Online; accessed 13-July-2023]. 2023 (cyt. na str. 7).

[148] Jason Williams, Antoine Raux i Matthew Henderson. „The Dialog State Tracking Challenge Series: A Review". W: *Dialogue & Discourse* (kw. 2016). URL: https://www.microsoft.com/en-us/research/publication/the-dialog-state-tracking-challenge-series-a-review/ (cyt. na str. 13).

[149] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes i Jeffrey Dean. *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. 2016. arXiv: 1609.08144 [cs.CL] (cyt. na str. 27).

[150] Zhaopeng Xing, Xiaojun Yuan, Dan Wu, Yeman Huang i Javed Mostafa. „Understanding Voice Search Behavior: Review and Synthesis of Research". W: *HCI International 2020 - Late Breaking Papers: Multimodality and Intelligence*. Red. Constantine Stephanidis, Masaaki Kurosu, Helmut Degen i Lauren Reinerman-Jones. Cham: Springer International Publishing, 2020, s. 305–320. ISBN: 978-3-030-60117-1 (cyt. na str. 8).

[151] Wayne Xiong, Jasha Droppo, Xuedong Huang, Frank Seide, Michael L. Seltzer, Andreas Stolcke, Dong Yu i Geoffrey Zweig. „Toward Human Parity in Conversational Speech Recognition". W: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 25.12 (2017), s. 2410–2423. DOI: 10.1109/TASLP.2017.2756440 (cyt. na str. 8).

[152] Yichong Xu, Chenguang Zhu, Shuohang Wang, Siqi Sun, Hao Cheng, Xiaodong Liu, Jianfeng Gao, Pengcheng He, Michael Zeng i Xuedong Huang. „Human Parity on CommonsenseQA: Augmenting Self-Attention with External Attention". W: *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*. Red. Lud De Raedt. International Joint Conferences on Artificial Intelligence Organization, lip. 2022, s. 2762–2768. DOI: 10.24963/ijcai.2022/383. URL: https://doi.org/10.24963/ijcai.2022/383 (cyt. na str. 8).

[153] Jiahui Yu, Chung-Cheng Chiu, Bo Li, Shuo-yiin Chang, Tara N. Sainath, Yanzhang He, Arun Narayanan, Wei Han, Anmol Gulati, Yonghui Wu i Ruoming Pang. „FastEmit: Low-Latency Streaming ASR with Sequence-Level Emission Regularization". W: *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Si-*

gnal Processing (ICASSP). Czer. 2021, s. 6004–6008. DOI: 10.1109/ICASSP39728.2021.9413803 (cyt. na str. 60).

[154] Yang Zhang, Evelina Bakhturina, Kyle Gorman i Boris Ginsburg. „NeMo Inverse Text Normalization: From Development to Production". en. W: *Interspeech 2021*. ISCA, sierp. 2021, s. 4468–4472. DOI: 10.21437/Interspeech.2021-1571. URL: https://www.isca-speech.org/archive/interspeech_2021/zhang21ga_interspeech.html (term. wiz. 20. 09. 2023) (cyt. na str. 8).

[155] Tomasz Ziętkiewicz. „Post-editing and rescoring of ASR results with edit operations tagging". W: *Proceedings of the PolEval 2020 Workshop*. Warsaw, Poland: Institute of Computer Science, Polish Academy of Sciences, 2020, s. 23–31. ISBN: "978-83-63159-29-0". URL: http://2020.poleval.pl/files/poleval2020.pdf#page=23 (cyt. na str. 11, 37).

[156] Tomasz Ziętkiewicz. „Punctuation Restoration from Read Text with Transformer-based Tagger". W: *Proceedings of the PolEval 2021 Workshop*. Warsaw, Poland: Institute of Computer Science, Polish Academy of Sciences, 2021, s. 54–60. ISBN: "978-83-63159-31-3" (cyt. na str. 11, 70).

[157] Tomasz Ziętkiewicz. „Tag and correct: high precision post-editing approach to speech recognition errors correction". W: *Proceedings of the 17th Conference on Computer Science and Intelligence Systems*. Red. Maria Ganzha, Leszek Maciaszek, Marcin Paprzycki i Dominik Ślęzak. T. 30. Annals of Computer Science and Information Systems. IEEE, 2022, s. 939–942. DOI: 10.15439/2022F168. URL: http://dx.doi.org/10.15439/2022F168 (cyt. na str. 10, 37).

Spis rysunków

Spis tabel