Podstawy strategii testowania języka Chatbota

Postęp technologii oraz rozwój sztucznej inteligencji spowodował pojawienie się na stronach internetowych firm, działających w sektorach handlowych oraz finansowych chatbotów. Wraz z tworzeniem tego oprogramowania idzie również jego testowanie oraz kontrola jakości. Niniejszy artykuł prezentuje podstawowe elementy oraz problemy testowania takich aplikacji jak chatbot. Celem artykułu jest przedstawienie ogólnej koncepcji testowania języka chatbotów.

1. Przeznaczenie chatbota

Najważniejszą kwestią podczas tworzenia chatbota jest określenie jego celu i przeznaczenie [1]. Zakres tematów, które potrafią obsłużyć chatboty (krócej bot) rośnie każdego dnia. Algorytmy uczące tego rodzaju oprogramowanie są ciągłym przedmiotem badań naukowych. Ta wiedza nie stoi w miejscu, ponieważ świat dąży do tego, aby boty mogły być zastosowany w jak największej liczbie gałęzi przemysłu czy handlu. Jednak boty nigdy w całości nie zastąpią ludzi. Bot może rozwiązać problem użytkownika, ale jedynie z pewnym prawdopodobieństwem. Ostatecznie człowiek musi podjąć decyzję, czy można zaufać botowi czy też nie. Wobec tego funkcjonowanie człowieka w biznesie czy handlu, na tym etapie rozwoju, nie może zostać całkowicie wyeliminowane przez boty.

Rodzaj interakcji chatbotów można podzielić na 3 kategorie [2]:

- Konwersacyjny - czyli imitujący rozmowę z prawdziwym człowiekiem,
- Scenariuszowy - składający się ze sztywno ustalonych pytań - w stylu formularzy,
- Mieszany - wykorzystujący oba podejścia.

Większość ludzi twierdzi, że bot powinien w 100% zastąpić człowieka. Obecna technologia nie jest jeszcze aż tak zaawansowana, by rozpoznać każdą intencję osoby komunikującej się z botem. Nie oznacza to jednak, że nie warto próbować. Tworzenie chatbota nie jest czynnością przewidywalną, tak jak to ma miejsce w tradycyjnym wytwarzaniu oprogramowania i to niezależnie od przyjętego modelu pracy. Wymaga dużego zaangażowania od twórców. Proces wytwórczy można rozpisać na kilka kroków:

1. Zidentyfikowanie celu i obszaru automatyzacji bota,
2. Rozpisanie możliwych scenariuszy, zapytań i odpowiedzi oraz kombinacji zapytań i odpowiedzi – ten etap jest najbardziej efektywny, jeżeli mamy dostęp do scenariuszy rozmów instytucji, dla której przeznaczony jest bot
3. Rozpoczęcie prac programistycznych,
4. Utworzenie rozmów z odpowiednią zawartością wyrażeń - zgodnie z zaprojektowanymi scenariuszami,
5. Testy bota
   - Testy automatyczne – sprawdzanie skuteczności kategoryzowania wprowadzonych fraz, testy wyuczonego kontekstu (możliwe rozpoznawanie zmiennych),
   - Testy rozmów,
   - Beta testy (np. prowadzone przez potencjalnych użytkowników),
6. Integracja chatbota z bazą danych lub z systemem informatycznym,

7. Utworzenie frameworka umożliwiającego zbieranie feedbacku oraz jego analizy.

W dalszej części artykułu skupimy się na zagadnieniach z punktu 5.

2. Testowanie danymi wejściowymi

Testowanie sterowane danymi to technika automatyzacji testów, która polega na umieszczeniu danych testowych i oczekiwanych wyników w tabeli lub arkuszu kalkulacyjnym, tak aby jeden skrypt mógł wykonać wszystkie testy z tabeli. Testowanie sterowane danymi jest często używane jako uzupełnienie narzędzi wykonywania testów takich jak narzędzia rejestrująco-odtwarzające [3]. W przypadku chatbota testowanie również będzie oparte na danych wejściowych, co więcej, kolejność wprowadzania tych zdań (rozumianych tutaj jako różne intencje użytkownika) będzie miała znaczenie, bo od niej zależą odpowiedzi bota.

Najważniejszą funkcjonalnością chatbota jest umiejętność rozmowy z użytkownikiem strony internetowej bądź aplikacji mobilnej. W tym przypadku test powinien wyglądać tak, jak rozmowa i może zawierać:

- wpisane zdania,
- krótkie wyrażenia,
- pojedyncze słowa,

z którymi bot powinien sobie poradzić udzielając poprawnej odpowiedzi. Słowo „poradzić" w tym przypadku nie zawsze będzie oznaczało, że użytkownik dostał wszystkie odpowiedzi takie, jakich oczekiwał. Poprawna klasyfikacja wprowadzonego zdania (odgadnięcie kategorii) oraz rozumienie kontekstu to najbardziej pożądane zachowanie bota. Jeśli dojdzie do sytuacji, że bot nie zrozumie człowieka, to przyczyną mogą być trzy podstawowe problemy:

- Bot nie rozpoznał wprowadzonej frazy,
- Bot nie potrafi się odnaleźć w kontekście prowadzonej rozmowy,
- Bot nie rozpoznał wprowadzonej frazy oraz kontekstu rozmowy jednocześnie.

Działanie bota powinno być tak zaprojektowane, aby nie wprowadzać w błąd osobę, która z nim pisze. Taka funkcjonalność powinna być dobrze przetestowana, ponieważ koszt pomyłki zawsze będzie większy niż koszt „przyznania się" do tego, że bot nie zrozumiał frazy albo kontekstu rozmowy. Każdy chatbot jest przeznaczony do rozmowy w konkretnej tematyce. Oprócz tego może pokusić się o rozmowę nazywaną ***small talk*** czyli pytania o luźnym charakterze np.:

- czy boty mają sny?
- czy znasz się na motoryzacji?
- co lubisz robić?
- jaka jest pogoda, itd.,

ponieważ użytkownicy zadają chatbotowi takie pytania z czystej ciekawości, aby dowiedzieć się jaka będzie jego odpowiedź. Taki stan rzeczy sprawił, że pojawiające się boty potrafią odpowiedzieć często półżartem, półserio na tego typu pytania. W dalszej części artykułu skupimy się głównie na testach języka, który jest wprowadzony do bota.

3. Myślenie skierowane na język naturalny i zdania

Frameworki lub platformy, które umożliwiają tworzenie botów, mogą być przeznaczone dla każdego z języków świata lub tylko dla wybranych. Języki na świecie możemy podzielić, najprościej mówiąc, na dwie kategorie: trudne i łatwe. Taki podział jest dość ogólny. Jednak są języki, które są naprawdę trudne dla bota (nie mówiąc o językach w których znaczenia ma ton wypowiedzi). Polski język jest jednym z najtrudniejszych języków na świecie. Trudność wynika z tego, że:

- Istnieje siedem przypadków, przez które możemy odmieniać rzeczowniki,
- Istnieją formy męskie, żeńskie oraz nijakie czasowników,
- Istnieją imiesłowy,
- Przymiotniki odmieniają się razem z rzeczownikami,
- Polski język ma swoje specyficzne znaki w alfabecie: ą,ś,ę,ż,ź,ć,ń,ó,ł.

Dla porównania, wyżej wymienione przykłady nie występują np. w języku angielskim. Takich przykładów, które występują w polskim języku a nie występują w innych można podać więcej. Przykładowo, jeśli chcemy zapytać bota o ubezpieczenie maszyn, to w polskim języku można to zrobić na wiele sposobów:

Przykład 1.

- *Mogę dostać informacje na temat ubezpieczenia maszyn?*
- *Mogę dostać informację na temat ubezpieczenia maszyny?*
- *Mogłabym dostać informację na temat ubezpieczenia maszyny?*
- *Mógłbym dostać informację na temat ubezpieczenia maszyny?*
- *Możesz powiedzieć coś o ubezpieczeniu maszyny?*
- *Możesz mi powiedzieć coś o ubezpieczeniu maszyny?*
- *Mógłbyś powiedzieć coś o ubezpieczeniu maszyny?*
- *Mógłbyś mi powiedzieć coś o ubezpieczeniu maszyny?*
- *Mógłbyś mi coś powiedzieć o ubezpieczeniu maszyny?*
- *Mógłbyś coś powiedzieć o ubezpieczeniu maszyny?*
- *Możesz mi coś powiedzieć o ubezpieczeniu maszyny?*
- *Powiedz mi o ubezpieczeniu maszyny!*
- *Powiedz mi coś o ubezpieczeniu maszyny!*
- *Powiesz mi coś o ubezpieczeniu maszyny?*
- *Powiesz mi coś o ubezpieczeniu maszyn?*
- *Powiesz mi coś o ubezpieczeniach maszyn?*
- *A powiesz mi coś o ubezpieczeniu maszyny?*
- *A powiesz mi coś o ubezpieczeniach maszyny?*
- *A może powiesz mi coś o ubezpieczeniu maszyn?*
- *A czy powiesz mi coś o ubezpieczeniu maszyn?*
- *Czy powiesz mi coś o ubezpieczeniu maszyn?*
- *…*

Wyżej wymienione przykłady nie wyczerpują sporej części pytań **w kategorii ubezpieczeń dla maszyn.** Kolejny problem z jakim muszą zmierzyć się twórcy bota w języku polskim to znaki specjalne np.

Przykład 2.

- *będę,*
- *wziął,*
- *ściął,*
- *wziął*
- *węzeł,*
- *…*

Takich przykładów również można podać mnóstwo. Ciekawymi przykładami dla bota są odmiany przymiotników z rzeczownikami np.:

- *biały pies,*
- *białego psa,*
- *białemu psu,*
- *z białym psem,*
- *o białym psie.*

Ciekawą cechą w polskim języku są też liczba mnoga, imiesłowy oraz formy czasowników (męskie, żeńskie i nijakie):

Przykład 3.

- *straciłem,*
- *straciłam,*
- *straciliśmy,*
- *straciłyśmy,*
- *straciłbym,*
- *straciło,*
- *straciłeś,*
- *stracił,*
- *straciła,*
- *straciliście,*
- *straciłyście,*
- *stracili,*
- *straciły,*
- *straciwszy.*

Język, który jest wymagany podczas tworzenia bota ma ogromne znaczenia w sensie dzielenia fraz (zdań, pytań oraz krótkich wyrażeń) na kategorie, wprowadzania zmiennych z wartościami (słowniki zmiennych np. nazwy Państw). Testy chatbota muszą być zaprojektowane tak, aby uwzględniały problemy występujące w danym języku. Co więcej, są języki na świecie, w których ton wypowiedzi zmienia znaczenia słowa! Jeśli postawimy dla chatbota wymaganie, aby potrafił ten ton rozpoznać, to jego twórcy mają ogromne pole do popisu.

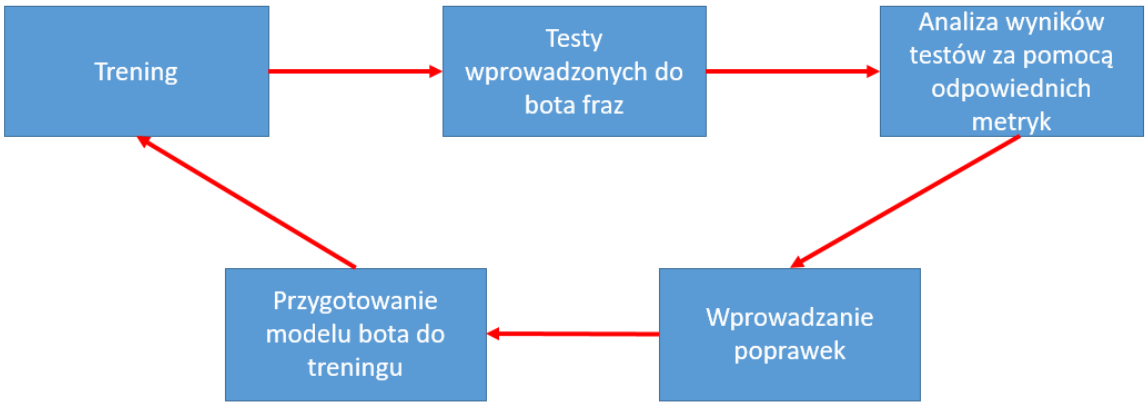
4. Testowanie języka, NLP

Testowanie języka, który jest obsługiwany przez bota ma jeden podstawowy problem – duża ilość danych wejściowych, nierealna do przetestowania manualnie. Twórcy bota mogą zastosować następujące metody:

- automatyzacja testów poprzez napisanie własnego frameworka oraz własnego mechanizmu generowania raportów z odpowiednimi metrykami,
- wykorzystanie do testów środowiska, które stanowi silnik bota,
- pomieszanie dwóch wyżej wymienionych strategii.

Najważniejsze jest to, aby środowisko testowe było użyteczne, dopasowane do wymagań bota, które chcemy testować oraz generowało raporty, które można analizować (analiza testów) i wyciągać z nich wnioski.

Dobrym pomysłem jest poddawanie testom wyrażeń wprowadzonych do bota przez jego twórców, w celu sprawdzenia, czy bot nauczył się poprawnie je klasyfikować. Po tym można cyklicznie udoskonalać proces treningu za pomocą parametrów, które oferuje sieć neuronowa albo dokonać zmian w kategoriach. Pokazuje to niżej przedstawiony obrazek:



Rysunek 1: Cykliczny schemat treningu i poprawy jakości klasyfikacji bota

Sieć neuronowa może być zdefiniowana tak, że wszystkie kategorie mają na siebie wpływ, stąd taki proces powinien być wykonywany po każdej zmianie zawartości fraz w kategoriach bota. Natomiast o samej jakości bota powinny decydować przyjęte przez twórców metryki, które mogą pomóc podjąć decyzję o dopuszczeniu pewnego zakresu rozmów do użytku ogólnego.

4.1 Testowanie literówek

Literówki, to często występujący rodzaj błędu, który występuje wszędzie tam, gdzie mamy do czynienia z pisaniem. W każdym języku bardzo łatwo można go popełnić, a zwłaszcza w polskim, ponieważ mamy specyficzne dla języka znaki. Przykładowo, w zdaniu *„Mogłabym dostać informację na temat*

ubezpieczenia maszyny?" można wygenerować olbrzymią liczbę literówek poprzez opuszczenie spacji, opuszczenie litery, braku polskiego znaku, wpisanie niepoprawnej litery itd. Twórcy bota powinni wiedzieć jak bardzo jest on odporny na takie pomyłki. Załóżmy, że istnieje framework, który sprawdza poprawność klasyfikacji kategorii wyrażenia wprowadzonego do bota. Takie testy przykładowo mogą mieć postać dwuwymiarowych wektorów:

- *Test_1: [Ubezpieczenie_maszyny: Mogłbym dosta informa cję natemat ubezpieczenia maszyny?],*
- *Test_2: [Ubezpieczenie_maszyny: Moglabym dostać infrmację na temat ubepieczenia maszny?],*
- *Test_3: [Ubezpieczenie_maszyny: Mogłabymdostać inormację na temat ubzpieczenia aszyny?],*
- *Test_4: [Ubezpieczenie_maszyny: Moglbym dostać informację na temat ubezpiecenia maszyny?],*
- *……..*
- *Test_n: [Ubezpieczenie_maszyny: Moglby dostać informację na tematubezpiecenia maszyny?],*

gdzie pierwsza współrzędna oznacza oczekiwaną kategorię, a druga wprowadzoną frazę – nasz test. Błędów związanych z literówkami można już w jednym zdaniu wygenerować bardzo dużo. Jeśli testy wykażą, że chatbot nie radzi sobie z literówkami, to jest to sygnał dla jego twórców, że należy wprowadzić do jego procesu uczenia rozumienie tych literówek. Jednym z rozwiązania takiego problemu może być wprowadzenie do danych uczących fraz z wyżej wspomnianymi literówkami. Jednak jest to czasochłonne.

Innym rozwiązaniem może być wykorzystanie generatora literówek, poprzez napisanie algorytmu ewolucyjnego z odpowiednimi prawdopodobieństwami mutacji dla znaków. Dla zestawu znaków charakterystycznych dla języka polskiego można wówczas ustawić wyższe prawdopodobieństwo mutacji znaku. Następnie plik z wygenerowanymi błędnymi frazami należy umieścić w danych treningowych bota i sprawdzić efekt testów literówek. Czynność tą można oczywiście powtarzać poprzez zmianę prawdopodobieństwa lub samego sposobu generowania błędów do momentu, aż testy dadzą pozytywny zdefiniowany przez twórców wynik w postaci miary liczbowej.
Aby utworzyć ewolucyjny algorytm generujący literówki, w pierwszej kolejności należy pomyśleć jak można podejść do generowania literówek. Można to zrobić np. za pomocą tabeli, która zawiera znak / znaki które mogą być literówkami oraz wartość prawdopodobieństwa mutacji znaku:

Litera	Literówki	Prawdopodobieństwo mutacji
spacja	brak spacji	0,2
ą	a	0,4
d	s,e,r,f	0,2
ć	c	0,4
k	j,i,o,l,","	0,1
ć	x,d,f,v," "	0,15
…	…	…

Algorytm ewolucyjny powinien działać tak, że przechodzi przez frazy umieszczone w danej kategorii, mutuje znaki, a następnie wynik mutacji zapisuje jako frazy, które można umieszczać w zestawie danych treningowych. Dodatkiem do algorytmu może być jeszcze losowe wstawianie spacji.

4.2 Testy na nietrenowanych frazach (dane walidacyjne). Testowanie poprawności klasyfikacji kategorii

Liczba wyrażeń, za którą stoją te same intencje użytkownika może być bardzo duża, np. 100 fraz. Twórcy chatbota nie są w stanie przewidzieć każdego jednego zdania, czy wyrażenia (niezależnie czy są one długie czy krótkie). Wobec tego dobrą praktyką jest sprawdzenie za pomocą ***automatycznych testów***, jak poradzi sobie bot podczas testów na treningowych danych.

Załóżmy, że mamy dwie kategorie dotyczące pytań o ubezpieczenie: *Ubezpieczenie_maszyny, Ubezpieczenie_pojazdu.* Załóżmy również, że w tych kategoriach mamy następujące dane treningowe:

- *Ubezpieczenie_maszyny:*
    - *Mogę dostać informacje na temat ubezpieczenia maszyn?*
    - *Mogę dostać informację na temat ubezpieczenia maszyny?*
    - *Mogłabym dostać informację na temat ubezpieczenia maszyny?*
    - *Mógłbym dostać informację na temat ubezpieczenia maszyny?*
    - *Możesz powiedzieć coś o ubezpieczeniu maszyny?*
    - *Możesz mi powiedzieć coś o ubezpieczeniu maszyny?*
    - *Mógłbyś powiedzieć coś o ubezpieczeniu maszyny?*
    - *Mógłbyś mi powiedzieć coś o ubezpieczeniu maszyny?*
    - *Mógłbyś mi coś powiedzieć o ubezpieczeniu maszyny?*
    - *Mógłbyś coś powiedzieć o ubezpieczeniu maszyny?*
    - *Możesz mi coś powiedzieć o ubezpieczeniu maszyny?*
    - *Powiedz mi o ubezpieczeniu maszyny!*
    - *Powiedz mi coś o ubezpieczeniu maszyny!*
    - *Powiesz mi coś o ubezpieczeniu maszyny?*
    - *Powiesz mi coś o ubezpieczeniu maszyn?*

- *Ubezpieczenie_pojazdu:*
    - *Mogę dostać informacje na temat ubezpieczenia pojazdów?*
    - *Mogę dostać informację na temat ubezpieczenia pojazdu?*
    - *Mogłabym dostać informację na temat ubezpieczenia pojazdu?*
    - *Mógłbym dostać informację na temat ubezpieczenia pojazdu?*
    - *Możesz powiedzieć coś o ubezpieczeniu pojazdu?*
    - *Możesz mi powiedzieć coś o ubezpieczeniu pojazdu?*
    - *Mógłbyś powiedzieć coś o ubezpieczeniu pojazdu?*
    - *Mógłbyś mi powiedzieć coś o ubezpieczeniu pojazdu?*
    - *Mógłbyś mi coś powiedzieć o ubezpieczeniu pojazdu?*
    - *Mógłbyś coś powiedzieć o ubezpieczeniu pojazdu?*

- *Możesz mi coś powiedzieć o ubezpieczeniu pojazdu?*
- *Powiedz mi o ubezpieczeniu pojazdu!*
- *Powiedz mi coś o ubezpieczeniu pojazdu!*
- *Powiesz mi coś o ubezpieczeniu pojazdu?*
- *Powiesz mi coś o ubezpieczeniu pojazdu?*

Każda kategoria zawiera po 15 fraz. Testy, które mogą zweryfikować poprawność klasyfikacji na niewyuczonych danych mogą mieć taką postać:

- *Test_1: [Ubezpieczenie_maszyny: Powiesz mi coś o ubezpieczeniach maszyn?],*
- *Test_2: [Ubezpieczenie_maszyny: A powiesz mi coś o ubezpieczeniu maszyny?],*
- *Test_3: [Ubezpieczenie_maszyny: A powiesz mi coś o ubezpieczeniach maszyny?],*
- *Test_4: [Ubezpieczenie_maszyny: A czy powiesz mi coś o ubezpieczeniu maszyn?],*
- *Test_5: [Ubezpieczenie_maszyny: A czy powiesz mi coś o ubezpieczeniu maszyn?],*
- *Test_6: [Ubezpieczenie_pojazdu: Powiesz mi coś o ubezpieczeniach pojazdów?],*
- *Test_7: [Ubezpieczenie_pojazdu: A powiesz mi coś o ubezpieczeniu pojazdów?],*
- *Test_8: [Ubezpieczenie_pojazdu: A powiesz mi coś o ubezpieczeniach pojazdów?],*
- *Test_9: [Ubezpieczenie_pojazdu: A czy powiesz mi coś o ubezpieczeniu pojazdów?],*
- *Test_10: [Ubezpieczenie_pojazdu: A czy powiesz mi coś o ubezpieczeniu pojazdów?],*

Jeżeli po wykonaniu testów okaże się, że nie wszystkie frazy zostały poprawnie rozpoznane, to takie frazy należy wprowadzić jako dane uczące oraz napisać nowe dane testowe. Taką czynność można powtarzać dotąd, aż bot zacznie poprawnie rozpoznawać dane testowe.

Ważne jest to, aby dane uczące w poszczególnych kategoriach były spójne, czyli zgodne z przeznaczeniem kategorii. Jeśli część fraz uczących pasujących do kategorii *Ubezpieczenie_maszyny* znajdzie się w innej kategorii, to jest to bardzo poważny błąd osób tworzących zawartość bota. Takie błędy należy wyłapywać jak najszybciej za pomocą dobrze skonstruowanych raportów lub po prostu do nich nie dopuszczać, bo wpływa to bardzo negatywnie na trenowanie bota.

4.3 Testowanie cech typowych dla konkretnego języka

Polski język ma wiele cech pod względem odmiany czasowników oraz rzeczowników, które nie zawsze występują w innych językach np. rodzaj czasownika w trzeciej osobie w liczbie pojedynczej, przypadki rzeczowników, odmiana przymiotników. Załóżmy, że pewien bot ma wprowadzone frazy z przykładu 1 oraz został poddany treningowi.

Aby sprawdzić, czy bot będzie w stanie rozpoznać pytania o ubezpieczenie maszyny ale nie te, które ma wprowadzone jako dane uczące, to należy napisać testy (w postaci fraz), które zawierają cechy typowe dla języka. Przykładowe takie testy dla form czasownika ***chcieć*** mogą mieć postać:

- *Test_1: [Ubezpieczenie_maszyny: Mój ojciec chciał się zapytać o ubezpieczenie maszyny?]*
- *Test_2: [Ubezpieczenie_maszyny: Moja siostra chciała dostać informację na temat ubezpieczenia maszyny?],*

- *Test_3: [Ubezpieczenie_maszyny: Moje dziecko chciało dostać informację na temat ubezpieczenia maszyny?],*

Testy dla odmiany rzeczownika ***maszyna*** przez przypadki mogą być napisane w sposób zaprezentowany niżej łącznie z odmianą przymiotnika ***nowy****:*

- *Test_1: [Ubezpieczenie_maszyny: Moja nowa maszyna musi mieć ubezpieczenie!]*
- *Test_2: [Ubezpieczenie_maszyny: Muszę ubezpieczyć moja nową maszynę!]*
- *Test_3: [Ubezpieczenie_maszyny: Chcę zapłacić ubezpieczenie mojej nowej maszynie.]*
- *Test_4: [Ubezpieczenie_maszyny: Potrzebuję ubezpieczyć nową maszynę.]*
- *Test_5: [Ubezpieczenie_maszyny: Jak to jest z ubezpieczeniem z nową maszyną?]*
- *Test_6: [Ubezpieczenie_maszyny: Mówię o nowej maszynie i o jej ubezpieczeniu.]*

Gdyby kategoria z maszynami zawierała pytania dotyczące maszyn (liczba mnoga) to doszłaby do testów odmiana przez przypadki dla liczby mnogiej rzeczownika „maszyna”. Można także pokusić się o napisanie testów uwzględniający ***czasy przeszły, przyszły i teraźniejszy,*** jeżeli będą one miały sens istnienia w danej kategorii bota. Pisanie testów poprzez ich odmianę przez przypadki polskiego języka generuje dodatkowo inne cechy testów np.: *interpunkcja, odmiana kilku przymiotników przez przypadki w jednym zdaniu, używanie synonimów dla czasowników zaczynających frazę*. Jeżeli testy wykażą, że bot potrafi rozpoznać nasze intencje, to znaczy że potrafi się wyuczyć odmian przymiotników oraz rzeczowników. Jeżeli testy wykażą niepoprawność klasyfikacji, to zdanie poddane testom należy wprowadzić jako dane uczące. Taki proces można powtarzać kilka razy, zwłaszcza dla polskiego języka.

4.3 Testowanie zmiennych

Kolejny problem, w którym muszą zmierzyć się twórcy chatbota to zmienne i ich wartości. Jeśli przeznaczeniem bota jest np. doładowanie telefonu, to musi on umieć rozpoznać kwotę, na jaką użytkownik chce doładować telefon oraz nazwę sieci komórkowej. Sam mechanizm rozpoznawania zmiennych tworzony jest przez programistów. Testerzy natomiast powinni wiedzieć, jak sprawdzić czy bot rozpoznaje zmienne poprawnie oraz jak pisać tego rodzaju testy. W tym przypadku automatyzacja jest również bardzo pożądana.

Jeśli chcemy, aby bot umiał odpowiedzieć na pytanie jaka jest pogoda w Warszawie, w Paryżu, czy w Płocku itd., to nie jest możliwe przetestowanie w rozsądnym czasie ręcznie wszystkich nazw miast z całego świata – to musi robić automat. Załóżmy, że bot potrafi zrozumieć pytania o pogodę umieszczone w kategorii *Weather.* Przykładowe frazy z tej kategorii mogą być takie:
jaka będzie dziś pogoda w Berlinie?

- *jaka dziś jest pogoda w Wałbrzychu?*
- *jaka dziś pogoda w Paryżu?*
- *prognoza pogody na jutro w Warszawie.*
- *Jaka będzie dziś pogoda w Rzymie?*
- *…*

Słowa zaznaczone żółtym kolorem to wartości zmiennej, która zawiera nazwy miast. Liczba miast na świecie jest bardzo duża. Twórcy bota powinni przewidzieć możliwość dodawania wartości do zmiennej za pomocą tabeli w bazie danych albo za pomocą pliku w odpowiednim formacie. Testy zmiennych można podzielić na dwa rodzaje:

- testowanie samej zmiennej,
- testowanie zmiennej w zdaniu.

Testy zmiennych przykładowo mogą być takie:

- *Test_1: [Weather: Moskwa],*
- *Test_2: [Weather: Moskwie],*
- *Test_3: [Weather: w Moskwie],*
- *Test_4: [Weather: Warszawa],*
- *Test_5: [Weather: Warszawie],*
- *Test_6: [Weather: Warszawy],*
- *Test_7: [Weather: Jaka jest pogoda w Warszawie],*
- *Test_8: [Weather: Jaka będzie dziś pogoda w Warszawie],*
- *Test_9: [Weather: Jaka będzie dziś pogoda w Szczecinie],*
- *Test_10: [Weather: Interesuje mnie pogoda w Katowicach],*
- *Test_11: [Weather: Interesuje mnie dzisiejsza pogoda w Elblągu],*
- *Test_12: [Weather: Interesuje mnie dzisiejsza pogoda w Elblągu],*
- *…*

W ciągu jednego dnia jedna osoba może napisać bardzo dużo takich testów automatycznych. Takie podejście pozwoli na bieżąco (po każdym treningu) kontrolować mechanizm wykrywania zmiennych przez bota. Sprawdzanie tego mechanizmu ręcznie jest niewykonalne przy założeniu, że bot jest trenowany raz dziennie. Sztuczna inteligencja jest na tyle nieprzewidywalna, że wprowadzenie danych niezwiązanych z kategorią *Weather* może popsuć wykrywanie tej lub innych kategorii, a co za tym idzie, pogorszyć wykrywanie zmiennych w zdaniu. Wewnątrz mechanizmu trenującego wszystko może być zależne od wszystkiego – stąd bierze się nieprzewidywalność bota po zakończonym treningu.

4.4 Wartość brzegowe w języku

Analiza wartości brzegowych to jedna z najbardziej popularnych technik testowania w przypadkach, gdzie dane można podzielić na ograniczone zbiory (skończone albo nieskończone). W przypadku bota zbiorem elementarnym będzie kategoria oznaczająca intencje użytkownika, a punktami zbioru będą wyrażenia (zdania albo krótkie oraz pojedyncze słowa). Brzegi w testowaniu to wartości, które mają charakter osobliwy. Teoretycznie wartości skrajne mogą wystąpić i aplikacja musi je uwzględniać, natomiast w praktyce występują o wiele rzadziej niż wartości nieosobliwe. Załóżmy, że chcemy napisać dane testowe (na których nie trenujemy bota), które są wartościami granicznymi dla kategorii *Ubezpieczenie_maszyny.* Osobliwymi testami będą:

(1) Najkrótsze wyrażenia jakie może wpisać użytkownik,

- *Test_1: [Ubezpieczenie_maszyny: Ubezpieczenie maszyny],*
- *Test_2: [Ubezpieczenie_maszyny: ubezpieczenie maszyn],*
- *Test_3: [Ubezpieczenie_maszyny: ubezpieczenia],*
- *Test_4:[Ubezpieczenie_maszyny: napisz o ubezpieczeniu],*
- *Test_5:[Ubezpieczenie_maszyny: napisz o ubezpieczeniach],*
- *Test_6:[Ubezpieczenie_maszyny: coś o ubezpieczeniach],*
- *Test_7:[Ubezpieczenie_maszyny: coś o ubezpieczeniu],*
- *…*

(2) Bardzo długie złożone zdania albo Więcej niż jedno zdanie,

- *Test_1: [Ubezpieczenie_maszyny: Dzień Dobry Panu, bardzo chciałbym się dowiedzieć czegoś o ubezpieczeniu maszyny, bo kupiłem sobie nową maszynę oraz zamierzam ją ubezpieczyć],*
- *Test_2: [Ubezpieczenie_maszyny: Dzień Dobry Panu, bardzo chciałbym się dowiedzieć czegoś o ubezpieczeniu maszyny, bo kupiłem sobie nową maszynę oraz dobrze by było ją ubezpieczyć],*
- *Test_3: [Ubezpieczenie_maszyny: Dzień Dobry, bardzo chciałbym się dowiedzieć czegoś o ubezpieczeniu maszyny. Kupiłem sobie nową maszynę oraz dobrze by było ją ubezpieczyć],*
- *Test_4: [Ubezpieczenie_maszyny: Dzień Dobry, bardzo chciałbym się dowiedzieć czegoś o ubezpieczeniu maszyny. Kupiłem sobie nową maszynę oraz dobrze by było ją ubezpieczyć],*
- *Test_5: [Ubezpieczenie_maszyny: Witam Pana bardzo serdecznie. Chciałbym się dowiedzieć czegoś o ubezpieczeniu maszyny. Kupiłem sobie nową maszynę oraz dobrze by było ją ubezpieczyć],*
- *Test_6: [Ubezpieczenie_maszyny: Witam Pana bardzo serdecznie. Chciałbym się dowiedzieć czegoś o ubezpieczeniu maszyny, ponieważ kupiłem sobie nową maszynę oraz dobrze by było ją ubezpieczyć],*
- *…*

Wyrażenia, które są bardzo długie, mogą sprawić kłopoty podczas klasyfikacji, ponieważ zawierają w sobie na ogół więcej niż jedną intencję. W takim przypadku do decyzji twórców bota należy to, czy wprowadzać takie zdania do danych treningowych. Jeśli stwierdzą oni, że takie początki rozmów będą występowały często, to próba nauczenia bota klasyfikacji tak długich fraz ma uzasadnienie, ale nie jest to proste.

Można podejść do tego problemu w inny sposób np. poprzez zaprojektowanie mechanizmu, który umożliwi rozpoczęcie rozmowy, który zaczyna się kilkoma kategoriami wprowadzanymi po kolei przez użytkownika. Takie podejście, w którym występuje więcej kategorii z krótszymi frazami jest lepsze z punktu widzenia klasyfikacji, bo dla bota będzie to prostsze do nauczenia się niż obszerne frazy umieszczone w jednej kategorii.

4.6 Przykładowe miary jakości klasyfikacji języka

Miary jakości dla chatbota można zdefiniować na różne sposoby. Te miary powinny być tak zaprojektowane, żeby osoby odpowiedzialne za jakość bota mogły stwierdzić, czy uczy się on

poprawnie wg przyjętych kryteriów czy też nie np. ponad 99 % poprawnie sklasyfikowanych fraz z danych uczących. Ale w takim przypadku należy sobie odpowiedzieć na jedno bardzo ważne pytanie:

Co to znaczy, że bot uczy się poprawie klasyfikować frazy?!

Odpowiedź wcale nie jest taka prosta. Załóżmy, że mamy zdefiniowane w chatbocie takie kategorie:

- *Ubezpieczenie_maszyny,*
- *Koszt_ubezpieczenia_maszyny,*
- *Rodzaj_ubezpiecznia_maszyny,*
- *Koszt_kredytu*

Jeżeli użytkownik wpisze zdanie: *Chcę zapłacić ubezpieczenie mojej nowej maszynie,* to nie znaczy, że bot sklasyfikuj je tylko do jednej kategorii. Klasyfikator powinien przyporządkować zdanie z bardzo dużą „wartością" tylko do jednej kategorii, natomiast może to zdanie również zostać przypisane do innych kategorii z małą wartością, np.:

Przykład 4.


```
ranking_kategorii: [
{
"wynik": 0.91662,
"nazwa": "Ubezpieczenie_maszyny"
},
{
" wynik ": 0.41624,
" nazwa ": "Koszt_ubezpieczenia_maszyny"
},
{
" wynik ": 0.30916,
" nazwa ": "Rodzaj_ubezpiecznia_maszyny"
},
{
" wynik ": 0.1759,
" nazwa ": "Koszt_kredytu"
}]
```


Wyrażenie *Chcę zapłacić ubezpieczenie mojej nowej maszynie* zostało sklasyfikowane jako *Ubezpieczenie_maszyny* z wartością 0.91662, natomiast druga w kolejce pasująca do tego zdania kategoria to *Koszt_ubezpieczenia_maszyny* z wartością 0.41624. Pozostałe dwie kategorie mają jeszcze mniejszą wartość dopasowania do wprowadzonego zdania. Załóżmy, że wartości przyporządkowania frazy do kategorii mieszczą się w przedziale (0;1). Wobec tego, patrząc na wyżej wyświetlone wyniki można stwierdzić, że **bot jest pewny co do sklasyfikowania tej frazy**, ponieważ różnica między pierwszą poprawną wartością klasyfikacji a drugą jest równa 0,50038. nie zawsze każda fraza będzie sklasyfikowana aż tak poprawnie, a to często ma miejsce podczas procesu tworzenia i treningu bota. Mogą pojawić się niepożądane sytuacja, takie jak:

(1) bot przyporządkował frazę niepoprawnie z dużą wartością:

ranking_kategorii: [
{
"wynik": 0.91662,
"nazwa": " Koszt_ubezpieczenia_maszyny"
},
{
" wynik ": 0.41624,
" nazwa ": " Ubezpieczenie_maszyny"
},
{
" wynik ": 0.30916,
" nazwa ": " Rodzaj_ubezpiecznia_maszyny"
},
{
" wynik ": 0.17519,
" nazwa ": "Koszt_kredytu"
}],

(2) bot przyporządkował frazę poprawnie z dużą wartością, ale drugą z kolei również przyporządkował z dużą wartością:

ranking_kategorii: [
{
"wynik": 0.91662,
"nazwa": "Ubezpieczenie_maszyny"
},
{
" wynik ": 0. 90599,
" nazwa ": "Koszt_ubezpieczenia_maszyny"
},
{
" wynik ": 0.30916,
" nazwa ": "Rodzaj_ubezpiecznia_maszyny"
},
{
" wynik ": 0.17519,
" nazwa ": "Koszt_kredytu"
}],

(3) bot przyporządkował frazę poprawnie z małą wartością:

ranking_kategorii: [
{
"wynik": 0.51518,
"nazwa": "Ubezpieczenie_maszyny"
},
{

" wynik ": 0. 41211,
" nazwa ": "Koszt_ubezpieczenia_maszyny"
},
{
" wynik ": 0.20916,
" nazwa ": "Rodzaj_ubezpiecznia_maszyny"
},
{
" wynik ": 0.17519,
" nazwa ": "Koszt_kredytu"
}].

Aby weryfikować jakość klasyfikacji, można wprowadzić parametr np. o nazwie *prog_pewnosci,* poniżej którego bot musi się „przyznać" użytkownikowi, że nie zrozumiał jego intencji, wysyłając odpowiedź. Jeżeli ustawimy *prog_pewnosci = 0.7,* to w przypadku sytuacji (3) bot zachowa się tak, jakby nie zrozumiał frazy. Jest to sygnał dla twórców bota, że należy wzmocnić mechanizm uczenia maszynowego albo dodać dane treningowe.

Sam *prog_pewnosci* jednak nie wystarczy. W przypadku sytuacji (2) nie wiemy, czy bota dobrze rozpoznał kategorię czy jest to tylko przypadek. Wobec tego dobrym pomysłem jest wprowadzenie funkcji, która liczy odległość pomiędzy dwoma pierwszymi kategoriami co do wartości klasyfikacji. Niech $K = \{K_1, K_2, \dots, K_n\}$ będzie zbiorem kategorii bota oraz niech $K_i = \{w_1, w_2, \dots, w_m\}$ będzie zbiorem fraz w kategorii K_i, gdzie $i \in \{1, \dots, n\}$. Dla każdego w_j definiujemy funkcję

$$f(w_j) = |w_j^1 - w_j^2|,$$

przy czym w_j^1 oznacza pierwszy najwyższy wynik klasyfikacji frazy w_j a w_j^2 oznacza drugi najwyższy wynik klasyfikacji frazy w_j, $j \in \{1, \dots m\}$. Niech dana będzie funkcja F określona wzorem

$$F(w_j) = \begin{cases} wynik\ klasyfikacji, & jeżeli\ f(w_j) > q \\ brak\ klasyfikacji, & jeżeli\ f(w_j) \leq q \end{cases},$$

gdzie q oznacza wartość różnicy pomiędzy dwiema pierwszymi sklasyfikowanymi kategoriami, poniżej którego bot przyznaje się, że nie jest pewny intencji użytkownika. Wartość parametru q powinna być dobrana tak, aby wyeliminować sytuację (2). Nie może być za duża ani za mała, bo nie rozwiąże problemu (2). Nie może też być zbyt duża, ponieważ doprowadzi do sytuacji, że bot cały czas będzie niepewny klasyfikacji. Wartość tego parametru powinna być ustalona na podstawie obserwacji zachowania bota, a to należy do decyzji jego twórców. Mając zdefiniowane miary liczące niepewność bota można napisać algorytm, który steruje zachowaniem bota:

1. Jeśli fraza została sklasyfikowana powyżej wartości prog_pewnosci oraz fraza pasuje do wyuczonego kontekstu rozmowy, to bot idzie kontekstem rozmowy,

2. Jeśli fraza została sklasyfikowana powyżej wartości prog_pewnosci ale fraza nie pasuje do wyuczonego kontekstu rozmowy, to bot wyświetla komunikat dla użytkownika, który o tej sytuacji odpowiednio informuje,

3. Jeśli fraza została sklasyfikowana poniżej wartości prog_pewnosci, to bot prosi, aby napisać zdanie innymi słowami,

4. Jeśli dwie najwyżej sklasyfikowane frazy są zbyt blisko siebie (wartość q), to bot prosi o napisanie frazy innymi słowami.

Twórcy bota powinni wiedzieć, jak „dobrze" wyuczona jest dana kategoria. Niech $m: K_i \rightarrow (0;1)$. Przy założeniach takich jak wyżej, miara, która o tym mówi może być przykładowo taka:

$$m(K_i) = \frac{1}{m}\sum_{j=1}^{m} K(w_j),$$

gdzie

$$K(w_j) = \begin{cases} wynik\ klasyfikacji, jeśli\ fraza\ została\ właściwie\ sklasyfikowana \\ 0\ w\ przeciwnym\ przypadku \end{cases}$$

Wartości miary M są unormowane dzięki stałej $\frac{1}{m}$. Jeśli twórcy bota zdecydują, że ta miara powinna być większa niż 0,9 i taka będzie, to oznacza, że bota jest dobrze wyuczony (pod warunkiem, że wszystkie frazy poddawane klasyfikacji zostały właściwie rozpoznane przez bota). Sama miara to nie wszystko, ponieważ może zdarzyć się taka sytuacja, że jedna lub dwie frazy zostaną sklasyfikowane, a pozostałe frazy będą sklasyfikowane poprawnie z dużą wartością. Oprócz samej miary, osoby wprowadzające dane uczące muszą wiedzieć, które frazy nie są poprawnie rozpoznawane przez bota za pomocą odpowiednio zdefiniowanego raportu. W sieciach neuronowych istnieje takie pojęcie jak confusion matrix [4]. Taką macierz można wykorzystać i umieścić w raporcie po treningu bota. Można zdefiniować analogiczną miarę jak miara m, ale dla wszystkich kategorii. Niech $M: K \rightarrow (0,1)$ będzie dana wzorem:

$$M(K) = \frac{1}{n \cdot m}\sum_{i=1}^{n}\sum_{j=1}^{m} K(w_j),$$

Wynikiem tej miary jest liczba z przedziału $(0;1)$, która mierzy jakość uczenia bota ogólnie, czyli dla wszystkich kategorii. Kolejną istotną miarą dla kategorii jest liczba fraz sklasyfikowanych poniżej wartości *prog_pewnosci.*

Wyżej opisane miary nie wyczerpują wszystkiego, co należy wiedzieć o zachowaniu bota. To najbardziej podstawowe miary. Jeśli dla bota prowadzone są testy na danych walidacyjnych (nie będących danymi treningowymi) np. dla literówek, to dla nich również można stosować wyżej opisane miary.

5. Podsumowanie

Celem napisania artykułu było przedstawienie najbardziej podstawowych problemów dotyczących testowania języka, dla którego projektowany jest bot. Opisane w poprzednim rozdziale to najbardziej podstawowe problemy oraz rodzaje testów, przez które powinien przechodzić bota nie jeden raz. Takie testowanie powinno trwać do momentu, aż twórcy uznają, że rozpoznawanie fraz testujących poszczególne rodzaje cech języka jest na tyle duże, że testy spełniły swoje zadanie - pomogły ustabilizować zawartość fraz w danym języku dla poszczególnych kategorii.

Należy jednak pamiętać o tym, że sama klasyfikacja nie tworzy chatbota, natomiast dochodzi jeszcze do tego kontekst rozmowy oraz zachowanie w wyjątkowych sytuacjach, z którym bot musi sobie poradzić. To wszystko razem, jeśli działa dobrze, tworzy dobrego bota.

6. Referencje:

[1] https://greenparrot.pl/blog/najwieksze-wady-chatbotow/

[2] https://highsolutions.pl/blog/wpis/co-ty-wiesz-o-chatbocie-czyli-techniczne-tajniki-chatbotow-o-ktorych-nie-miales-pojecia

[3] http://testerzy.pl/slownik/testowanie-sterowane-danymi

[4] https://en.wikipedia.org/wiki/Confusion_matrix

Marek Żukowicz jest absolwentem matematyki na Uniwersytecie Rzeszowskim. Jest testerem oprogramowania w firmie Ailleron. Jego zainteresowania skupiają się wokół testowania, matematyki, AI, zastosowania modeli matematycznych w procesie testowania.