Gramatyka aplikacji głosowej

Proces tworzenia gramatyki aplikacji głosowej składa się z następujących etapów:

1. Zdefiniowanie dialogu
2. Określenie pozycji informacyjnych i zdefiniowanie slotów
3. Zaprojektowanie podpowiedzi
4. Wybór typu gramatyki
5. Określenie odpowiedzi rozmówcy
6. Określenie bazowych fragmentów gramatyki
7. Określenie wypełniających fragmentów gramatyki
8. Napisanie kodu dla gramatyki
9. Dodanie poleceń w języku naturalnym

Ad1. Dialog należy zdefiniować przed rozpoczęciem pisania gramatyki **-** dialog określa jakie gramatyki należy napisać. Dla aplikacji komercyjnej dialog jest zazwyczaj definiowany w formalnym dokumencie specyfikacji dialogu, ale można użyć dowolnego rodzaju dokumentacji sensownej dla danego projektu. Jako minimum, należy odpowiedzieć na następujące pytania, determinujące kształt i treść tworzonych gramatyk:

- Jakie informacje są wymagane do realizacji zadania?
- W jakiej kolejności informacje będą wymagane?
- Czy dialog będzie wymagał pojedynczych informacji w określonej kolejności – dialog sterowany - czy pozwoli na wprowadzenie w dowolnej kolejności kilku informacji w jednej wypowiedzi i w razie konieczności poprosi o brakujące dane – dialog o przemiennej inicjatywie?

Ad2. Normalnie należy użyć jednego slotu dla każdej danej (pozycji informacyjnej). Slot przyjmuje wartość określonego typu.

Na przykład, w aplikacji informacji o lotach, prawdopodobnie będzie potrzebne pozyskanie nazw dwóch miast (startu i przeznaczenia), daty i czasu, a następnie potwierdzenie ważności zebranych informacji (pytanie tak/nie), w sumie pięć danych. Można również określić format i typ tych danych (tablica 1).

Tablica 1

Dana	Nazwa slotu	Format	Typ
miasto #1	start	kod 3-literowy	string
miasto #2	meta	kod 3-literowy	string
data	data	[miesiąc dzień]	struktura
czas	czas	0 - 2359	integer
tak/nie	potwierdzenie	„tak” lub „nie”	string

Ad3. Podpowiedzi najlepiej zaprojektować przed napisaniem gramatyki, ponieważ ich sformułowanie może znacznie wpłynąć na treść odpowiedzi rozmówcy. Twórca gramatyki musi przewidzieć te odpowiedzi - zmiany podpowiedzi podczas tworzenia gramatyki spowodują konieczność wielu poprawek.
Uwaga: podstawowe dane, takie jak nazwy miast i listy nazw, mogą być opracowane wcześniej, przed zakończeniem projektu dialogu.

Na przykład, aby otrzymać informacje o locie, podpowiedzi i sloty mogą być takie jak w tablicy 2.

Tablica 2

Podpowiedź	Slot
W którym mieście wylot?	start
W którym mieście lądowanie?	meta
Jaka jest data wylotu?	data
Jaka jest godzina wylotu?	czas
Czy wylot to <start> lądowanie to <meta> data to <data> i godzina to <czas>? Czy wszystko dobrze?	potwierdzenie

Jeśli są dodatkowe podpowiedzi dotyczące reakcji na błąd lub żądanie pomocy, które mogą bezpośrednio poprzedzać rozpoznawanie, należy napisać je również i wziąć je pod uwagę podczas pisania gramatyki.

Uwaga: Te poprzedzające podpowiedzi są właściwe dla dialogu kierowanego. Dialog o przemiennej inicjatywie może zacząć się od pytania "Gdzie chcesz podróżować" lub "Jak mogę pomóc?", a następnie stawiać bardziej szczegółowe pytania, aby uzyskać brakujące dane. Dużo trudniej przewidzieć zakres odpowiedzi na pytanie otwarte - to sprawia, że gramatyka jest trudniejsza do napisania. Jednakże dialog o przemiennej inicjatywie jest bliższy interakcji między ludźmi i może zmniejszyć liczbę interakcji użytkownik - system. Projektant decyduje, czy dialog wymusza jedynie proste, czy bardziej złożone odpowiedzi.

Ad4. Gramatykę można napisać przy użyciu następujących formatów:

- GSL (Grammar Specification Language) firmy Nuance
- XML W3C
- SLM (Statistical language model)

Formaty GSL i XML dostarczają dwóch różnych syntaktyk reprezentowania tej samej gramatyki bazowej. Formaty te są znane jako gramatyki probabilistyczne o skończonej liczbie stanów. Są użyteczne, gdy podpowiedzi w aplikacji są wystarczające, aby ograniczyć odpowiedź użytkownika. Ogólnie rzecz biorąc, GSL i XML są odpowiednie dla większości aplikacji.

W przeciwieństwie do gramatyki GSL lub XML, gramatyka SLM nie jest pisana odręcznie, ale uzyskiwana w czasie procesu uczenia ze zbioru przykładów, które modelują wypowiedzi użytkownika. Aby pozyskać gramatykę SLM, należy zbiór przykładów (i opcjonalnie właściwe dla zastosowania słownictwo) przetworzyć za pomocą odpowiedniego narzędzia, które wyznacza prawdopodobieństwa wypowiedzi (fraz). Gramatyka SLM jest odpowiednia do rozpoznawania mowy w wypowiedziach spontanicznych, w szczególności, gdy poziom zdarzeń pozagramatycznych (out-of-grammar rate) dla gramatyk GSL lub XML jest wysoki.

Rozważmy aplikację, która wymaga otwartej podpowiedzi, jak "Proszę podać charakter problemu". Nie tylko odpowiedź użytkownika jest wysoce zmienna i trudna do przewidzenia, ale może również zawierać restarty, przerwy (um i uh) i zdania niegramatyczne. Gramatyka SLM byłaby odpowiednia dla takiej aplikacji. Gramatyki SLM pozwalają użytkownikom na swobodne wypowiedzi i interpretują je zgodnie z zasadami przetwarzania języka naturalnego bez

konieczności pisania skomplikowanych reguł gramatycznych, obejmujących całe zdanie.

Uwaga: gramatyka SLM nie jest dostępna w standardzie VXML.

Ad5. Po zaprojektowaniu podpowiedzi, można przewidzieć dokładniej jak odpowiedzą osoby dzwoniące. Zgodnie z poprzednimi rozważaniami, istnieją dwa rodzaje odpowiedzi, które są najbardziej popularne:

- wiadomość jako taka,
- dosłowna odpowiedź na sformułowane pytanie.

Należy również uwzględnić fakt, że ludzie na początku się wahają, a czasem mówią "proszę" na końcu wypowiedzi. Nie należy próbować pokryć całego mówionego języka wypowiedzi (np. języka polskiego). Zmniejszenie rozmiaru gramatyki poprawi szybkość i dokładność systemu.

Gramatyki powinny być dostosowane do specyfiki zadania i ograniczać rozpoznawanie do jego zakresu. Podczas gdy wypowiedzi pozagramatyczne mogą być poważnym źródłem błędów, próba pokrycia całości obniża dokładność. W zadaniach, gdzie poziom zdarzeń poza gramatycznych jest problemem należy rozważyć gramatyki SLM.

Tablica 3 pokazuje przewidywane odpowiedzi użytkownika na wcześniej sformułowane podpowiedzi.

Tablica 3

Z którego miasta wylot?	
Wylot z Warszawy	[odpowiedź dosłowna]
Warszawa	[nazwa miasta]
Z Warszawy	[nazwa miasta z przyimkiem]
Mhm, z Warszawy	[początkowe niezdecydowanie]
Warszawa, proszę	[końcowe „proszę”]
Wylatuję z Warszawy	[inne możliwości]
Chciałbym wylecieć z Warszawy	
W którym mieście lądowanie?	
Lądowanie w Poznaniu	[odpowiedź dosłowna]
Poznań	[nazwa miasta]
W Poznaniu	[nazwa miasta z przyimkiem]
Mhm, w Poznaniu	[początkowe niezdecydowanie]
W Poznaniu, proszę	[końcowe „proszę”]
Chciałbym wylądować w Poznaniu	[inne możliwości]
Zamierzam w Poznaniu	
Jaka jest data wylotu?	
Data wylotu to dwunasty czerwca	[odpowiedź dosłowna]
Dwunasty czerwca	[tylko data]
Eee, dwunastego czerwca, proszę	[niezdecydowanie + ”proszę”]
Wylot dwunastego czerwca	[inne możliwości]
Chciałbym wylecieć dwunastego czerwca	
Jaka jest godzina wylotu?	
Godzina wylotu szósta	[odpowiedź dosłowna]
Szósta	[tylko godzina]
O szóstej, proszę	[końcowe ”proszę”]
Wylatuję o szóstej rano	[inne możliwości]
Chciałbym wylecieć o szóstej	
Wylot o godzinie szóstej	
Czy wylot to <start> lądowanie to <lądowanie> data to <data> i godzina to <czas>? Czy wszystko dobrze?	
Tak	[tylko potwierdzenie]
Nie	[tylko zaprzeczenie]
Tak, wszystko dobrze	[odpowiedź dosłowna]
Wszystko dobrze	[odpowiedź dosłowna]
Nie, niedobrze	[odpowiedź dosłowna]
Tak jest	
OK	

Ad6. Gramatyka zwykle składa się z części bazowej (core portion), która zawiera najważniejsze słowa zawierające najważniejsze informacje - takie jak miasta, daty i czas – oraz części wypełniającej, która zawiera dodatkowe określenia takie jak "chciałbym" lub "proszę".

Część bazowa jest wielokrotnie używana, więc ma sens zdefiniowanie subgramatyki lub reguły gramatycznej – mniejszej gramatyki wykorzystywanej w budowaniu hierarchii w większych gramatykach – opisującej tylko bazowe

części gramatyki. Informacja, która odnosi się do szczegółowej gramatyki może być dodana na wyższym poziomie tworzonej gramatyki.
W przykładzie informacji o lotach, subgramatyki bazowe powinny opisywać miasta, daty, godziny i potwierdzenie.

Ad7. Wypełniające fragmenty gramatyki w znacznej mierze zależą od sformułowań podpowiedzi. Po rozważeniu odpowiedzi rozmówcy, jak opisano w punkcie „określenie odpowiedzi rozmówcy", następuje zastępowanie bazowej części każdej wypowiedzi, w wykazie przewidywanych zwrotów, nazwami gramatyki bazowej. Część pierwotnej odpowiedzi, która pozostaje po wymianie jest, najprawdopodobniej, wypełniającymi fragmentami gramatyki.

W przykładzie informacji o lotach, można zastosować regułę gramatyczną MIASTO i DATA, prowadząc do następujących rodzajów przekształconych fraz:

Z którego miasta wylot?
wylot z MIASTO
MIASTO
z MIASTO
mhm, z MIASTO
MIASTO, proszę
wylatuję z MIASTO
chciałbym wylecieć z MIASTO

Jaka jest data wylotu?
data wylotu to DATA
DATA
eee, DATA, proszę
wylot DATA
chciałbym wylecieć DATA

Ad8. Do napisania gramatyki można zastosować format GSL, XML lub SLM. Ponieważ w powyższym przykładzie obydwie gramatyki (miasto wylotu i data) są dość proste oraz odpowiedzi użytkownika mogą być łatwo ograniczone, do napisania gramatyki wystarczą formaty GSL i XML.

Zakładając, że MIASTO i DATA są subgramatykami zdefiniowanymi w innym miejscu, kod GSL może być następujący:

```
MIASTO_WYLOTU [
(wylot z MIASTO)
MIASTO
(z MIASTO)
(mhm, z MIASTO)
(MIASTO, proszę)
(wylatuję z MIASTO)
(chciałbym wylecieć z MIASTO)
]

DATA_WYLOTU [
(data wylotu to DATA)
DATA
(eee, DATA, proszę)
(wylot DATA)
```

```
(chciałbym wylecieć DATA)
]
```

Można wstawić wypełniacze do gramatyki:

```
MIASTO_WYLOTU (?Miasto_Wypelniacz MIASTO ?please)

DATA_WYLOTU (?Data_Wypelniacz DATA ?please)
```

Gdzie Miasto_Wypelniacz i Data_Wypelniacz są subgramatykami, zdefiniowanymi w innym miejscu np.:

```
Miasto_Wypelniacz [
(wylot z)
 (z)
(mhm, z)
 (wylatuję z)
(chciałbym wylecieć z)
]
```

Pojawiły się dodatkowe możliwości wyrażania, np. fraza „mhm, Kraków proszę”.

:

Ad9. Pokazane to zostanie na poniższym przykładzie, w którym:

- c i d są zmiennymi
- wyrażenia MIASTO:c i DATA:d przypisują zmiennym c i d wartości zwrócone przez subgramatyki MIASTO i DATA
- wyrażenia $c i $d są odwołaniami do wartości odpowiednich zmiennych
- wyrażenia {<start $c>} i {<data $d>} przypisują slotom start i data wartości przechowywane w zmiennych c i d.

```
MIASTO_WYLOTU [
(wylot z MIASTO:c)
MIASTO:c
(z MIASTO:c)
(mhm, z MIASTO:c)
(MIASTO:c, proszę)
(wylatuję z MIASTO:c)
(chciałbym wylecieć z MIASTO:c)
]
{<start $c>}
DATA_WYLOTU [
(data wylotu to DATA:d)
DATA:d
(eee, DATA:d, proszę)
(wylot DATA:d)
(chciałbym wylecieć DATA:d)
]
{<data $d>}
```