

Realizacja prostego silnika reguł walidacyjnych przy użyciu technik NLP

Mariusz Wójcik

12 czerwca 2019

1 Początki

Jakiś czas temu miałem okazję obejrzeć film, który zrobił na mnie ogromne wrażenie. „*Arrival - Nowy Początek*” w reżyserii Denisa Villeneuve’a - to obraz niezwykle. Porusza on problematykę szerokokopijetęj, wielopłaszczyznowej komunikacji (a czasem skutków jej braku) . W niesamowicie sugestywny i obrazowy sposób pokazuje mechanizm kształtowania się podstaw wspólnego języka i nawiązywania kontaktu. Proces stopniowego budowania uwspólnionych modeli pojęciowych prowadzący do porozumiewania się tym samym językiem wydał mi się tak logiczny i uporządkowany, że sprawiał wrażenie niemal algorytmicznego... Pamiętam swoją myśl, że skoro możliwe jest tak precyzyjne określenie reguł stojących u podstaw nawiązania skutecznej komunikacji, to droga do zbudowania inteligentnych maszyn porozumiewających się z nami „ludzkim” językiem wydaje się już bardzo krótka.

Do niedawna wydawało mi się, że porozumiewanie się językiem naturalnym jest domeną przynależną wyłącznie człowiekowi. Miałem poczucie, że prace nad komputerowym przetwarzaniem języka naturalnego mają wymiar wyłącznie akademicki. Okazało się jednak, że dynamiczny rozwój algorytmów sztucznej inteligencji i przetwarzania maszynowego dotknął również tej dziedziny. Gdzieś na styku matematyki, informatyki i lingwistyki wykształciła się dziedzina, która funkcjonuje jako *NLP* (*ang. natural language processing*).

Techniki NLP koncentrują się na analizie, przekształcaniu i generowaniu języka naturalnego. Dzięki nim komputery nabywają umiejętności nie tylko analizy tekstu, ale również nauki i wyciągania wniosków. Dają one możliwość analizy nie tylko składni zdań, ale również doszukiwania się ich znaczeń i ukrytych pomiędzy słowami intencji. Czasami uświadamiam sobie że to wszystko razem brzmi jak czysta fantastyka. Bo jak niby sens, znaczenie i intencje można przeliczyć na liczby i twardo zakotwiczyć w dziedzinie algebry liniowej ?

Tajemnicy tej uchyla jedna z najciekawszych książek, jaką miałem przyjemność ostatnio czytać, mianowicie „*Natural Language Processing in Action*”. Jest to bardzo przystępnie napisany przewodnik, dzięki któremu łatwiej oswoić się z podstawowymi prawami rządzącymi światem NLP. Pozycja nie traktuje o rzeczach najłatwiejszych, a mimo to czyta się ją z dużą przyjemnością.

Z teorią często jest tak, że w którymś momencie chciałoby się ją zobaczyć w praktyce. Z tej potrzeby zrodził się pomysł na aplikację, którą możnaby zrealizować przy użyciu technik i algorytmów NLP. Przyszedł mi do głowy generator kodu aplikacji, który byłby w stanie przekształcić tekst napisany językiem zbliżonym do naturalnego bezpośrednio do kodu wykonywalnego. Oczywiście zakładałam że tego typu rozwiązanie miałoby zastosowanie do jakiegoś ściśle określonego aspektu działającej aplikacji, np. walidacji dokumentu, czy sprawdzania reguł poprawności modelu dziedziny. I tak właśnie powstał mój miniprojekt, którego celem jest zobaczenie o co tak naprawdę chodzi z tym NLP . :) . Zapraszam do zapoznania się z założeniami i otrzymanymi wynikami.

Mam świadomość, że jeśli chodzi o NLP, jestem na początku drogi. Nie mogę powiedzieć nawet tego, że udało mi się zrobić jeden krok, ale wiem jedno... podróż zapowiada się naprawdę imponująco...

2 Realizacja

Do realizacji moich założeń wybrałam napisaną w Javie bibliotekę *Apache OpenNLP*. Dostarcza ona narzędzi realizujących wiele aspektów przetwarzania języka naturalnego. W moim projekcie skupię się technice nazywanej *Named Entity Recognition (NER)*.

Polega ona na rozpoznawaniu w tekście określonych bytów nazwanych. Najczęściej są to imiona, nazwiska, nazwy własne itp. W moim przypadku chciałbym stworzyć własny model, który zostanie przyuczony do rozpoznawania poszczególnych elementów konstrukcji reguły walidacyjnej (takich jak słowa kluczowe rozpoczynające i kończące bloki, operatory, akcje i ich parametry).

Żeby to osiągnąć konieczne jest przygotowanie odpowiednio opisanej próbki uczącej, a następnie wykorzystanie jej do treningu modelu.

3 Abstrakcyjny model reguły

Przygotowanie próbki rozpocznę od opracowania schematu reguły.

Na początek wypiszę sobie kilka przykładowych reguł walidacyjnych.

1. Jeśli `wiek_pacjenta` jest większy od 18 wtedy zgłoś błąd „Pacjent jest osobą dorosłą.”, w przeciwnym wypadku wyświetl komunikat „Pacjent został zakwalifikowany do leczenia pediatrycznego.”.
2. Jeśli `data_kwalifikacji` jest mniejsza od '01-01-2019' wtedy zgłoś wyjątek „Data sprzed roku 2019.”, w przeciwnym wypadku sprawdź regułę RS-001.
3. Gdy `saldo_rachunku` jest większe od 100 oraz `saldo_rachunku` jest mniejsze niż 1000 wtedy wyświetl komunikat „Saldo rachunku jest prawidłowe.”, w przeciwnym razie zgłoś błąd „Nieprawidłowe saldo rachunku”.
4. Jeśli `data_teraz` jest nie większa niż `data_ważności` wyświetl komunikat „Wniosek jest aktualny.” w przeciwnym wypadku zgłaszaj błąd „Wniosek utracił ważność”.

Przyjmuję uproszczenie, że każda rozpoznawana reguła składała się będzie z trzech wyróżnialnych bloków:

$$\underbrace{WARUNKI}_{\text{Warunki}} \underbrace{AKCJA_TAK}_{\text{Akcja Tak}} (\underbrace{AKCJA_NIE}_{\text{Akcja Nie}})?$$

Poszczególne bloki oddzielone będą od siebie słowami kluczowymi oznaczającymi rozpoczęcie i zakończenie bloku.

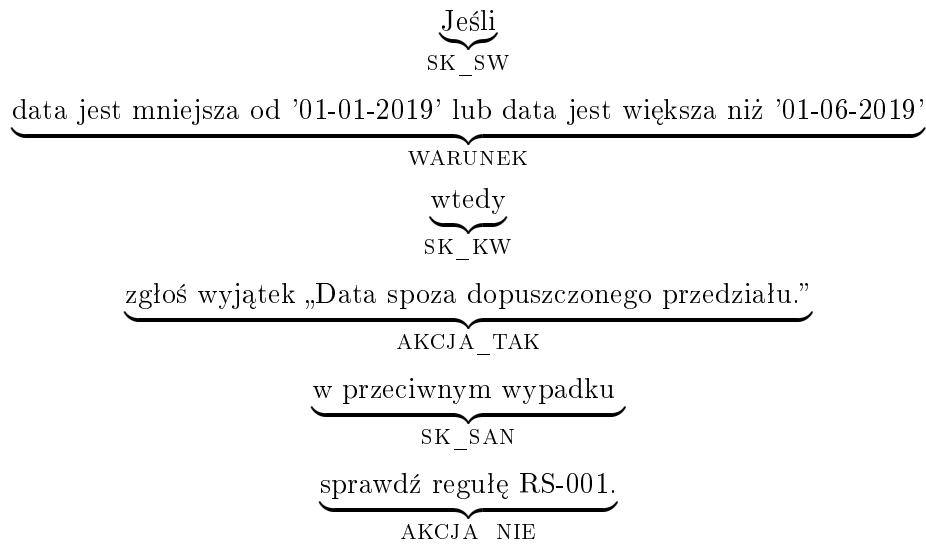
W celu ich wyróżnienia wprowadzam następujące oznaczenia:

1. SK_SW - Start sekcji warunku
2. SK_KW - Koniec sekcji warunku
3. SK_SAN - Start sekcji akcji wykonywanej przy niespełnionym warunku

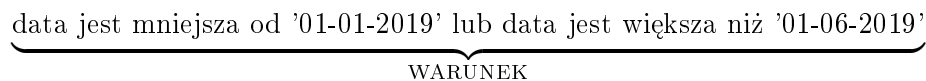
Schemat reguły przyjmuje następującą postać:

$$\underbrace{SK_SW}_{\text{SK_SW}} \underbrace{WARUNKI}_{\text{WARUNKI}} \underbrace{SK_KW}_{\text{SK_KW}} \underbrace{AKCJA_TAK}_{\text{AKCJA_TAK}} (\underbrace{SK_SAN}_{\text{SK_SAN}} \underbrace{AKCJA_NIE}_{\text{AKCJA_NIE}})?$$

Rzut oka na przykład:



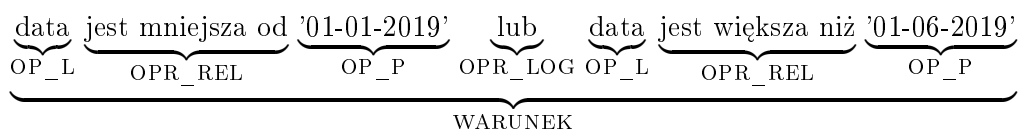
Ponieważ kluczowe jest właściwe rozpoznanie sekcji warunku, chciałbym wyłączyć go przed nawias i przez chwilę skupić się wyłącznie na nim.



Na początek trzeba zauważyć, że powyższa sekcja składa się z dwóch niezależnych wyrażeń warunkowych połączonych operatorem logicznym *lub*. Każdy z pojedynczych warunków składa się z kolei z operatora relacyjnego (*jest mniejsza*, *jest większa*), oraz z dwóch operandów (*lewego i prawego*). Wprowadzam więc następujące oznaczenia:

1. OP_L - Operand lewy
2. OPR_REL - Operator relacyjny
3. OP_P - Operand prawy
4. OPR_LOG - Operator logiczny

Po podstawieniu, przykładowy warunek można



Zatem zapis symboliczny sekcji warunkowej będzie wyglądał następująco:



Po dokonaniu podstawienia w sekcji *WARUNKI* abstrakcyjny model reguły przyjmie następującą postać:

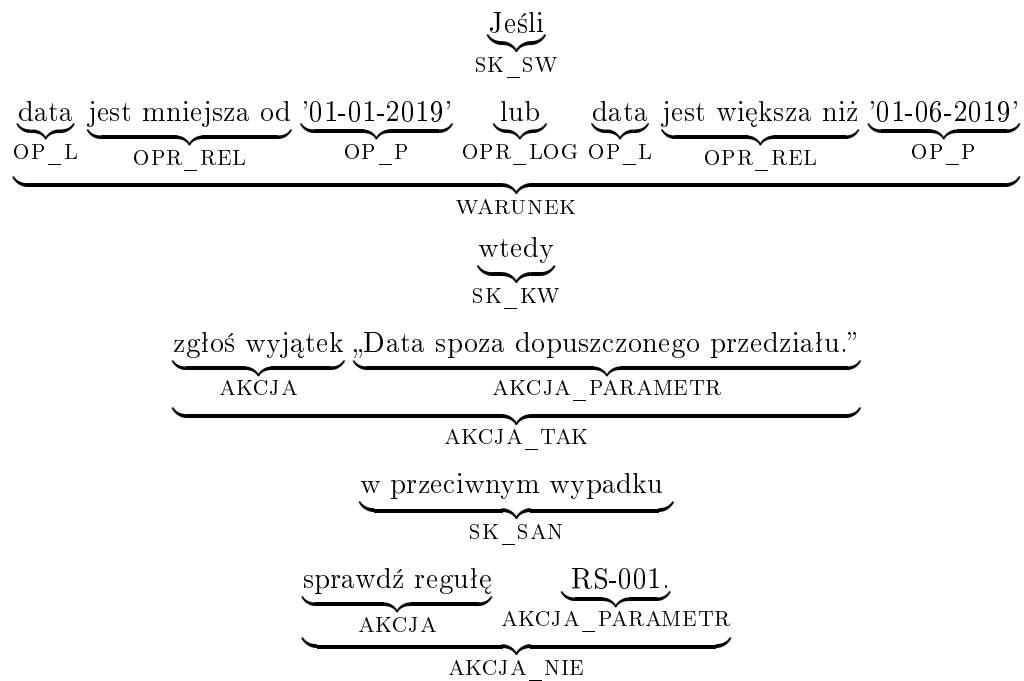
$$\begin{array}{c}
 \underbrace{SK_SW} \\
 \underbrace{OP_L\ OPR_REL\ OP_P\ (OPR_LOG\ OP_L\ OPR_REL\ OP_P)?}_{WARUNEK} \\
 \underbrace{SK_KW} \\
 \underbrace{AKCJA_TAK} \\
 (\underbrace{SK_SAN}\ \underbrace{AKCJA_NIE})?
 \end{array}$$

Jeśli chodzi o akcje, to sprawa wydaje się prostsza, bo każda z nich składa się z części mówiącej o tym co ma być zrobione (AKCJA), oraz z jakim parametrem ma być wykonane (AKCJA_PARAMETR). Ostatecznie więc, po wykonaniu podstawienia nasz model przyjmie następującą, ostateczną formę:

$$\begin{array}{c}
 \underbrace{SK_SW} \\
 \underbrace{OP_L\ OPR_REL\ OP_P\ (OPR_LOG\ OP_L\ OPR_REL\ OP_P)?}_{WARUNEK} \\
 \underbrace{SK_KW} \\
 \underbrace{AKCJA\ AKCJA_PARAMETR}_{AKCJA_TAK} \\
 (\underbrace{SK_SAN}\ \underbrace{AKCJA\ AKCJA_PARAMETR})?_{AKCJA_NIE}
 \end{array}$$

1. SK_SW - Start sekcji warunku
2. SK_KW - Koniec sekcji warunku
3. SK_SAN - Start sekcji akcji wykonywanej przy niespełnionym warunku
4. OP_L - Operand lewy
5. OPR_REL - Operator relacyjny
6. OP_P - Operand prawy
7. OPR_LOG - Operator logiczny

I jeszcze ostatnie spojrzenie na przykład:



Tak określony model może być podstawą do wygenerowania odpowiednio oznakowanej próbki uczącej.