

System do automatycznej klasyfikacji recenzji filmowych

Projekt z przedmiotów SAG i WEDT

Adam Kowalewski

Piotr Hachaj

Paweł Rybak

I. Wstęp

Założeniem projektu było stworzenie systemu agentowego, potrafiącego odzwierciedlić recenzję napisaną językiem naturalnym, w postaci oceny liczbowej. Przy tworzeniu systemu przyjęliśmy następujące założenia:

- system przyjmuje recenzje w języku polskim,
- tekst będzie przetwarzany przy użyciu technik NLP,
- zastosowana zostanie prezentacja tekstu w postaci *bag of words*,
- tekst będzie przetwarzany przez niezależnie przez kilka instancji algorytmów ML,
- uszkodzenie pojedynczego aktora klasyfikującego nie zatrzyma pracy systemu,
- projekt zostanie stworzony przy użyciu języka *Scala* i bibliotek *Akka* oraz *Spark*.
- zostanie położony nacisk na zrównoleganie obliczeń między procesorami, a nawet między maszynami

II. Pozyskanie danych uczących

Dane uczące do projektu zostały pobrane z następujących polskich serwisów internetowych o tematyce filmowej:

- filmweb.pl,

- filmaster.com,
- filmawka.pl,

Niestety, żadna z wymienionych witryn nie udostępnia wyspecjalizowanego API do integracji, dlatego dane musiały zostać pozyskane przy użyciu technik *web scrapingu*. Stworzone zostały do tego celu specjalne skrypty w języku Python, które przy użyciu listy dostępnych filmów, udostępnianej przez każdy z tych serwisów, pobierały dla każdego z nich plik html z recenzją oraz wyodrębniały z nich treść recenzji i ocenę liczbową nadaną przez autora. Udało się w ten sposób pobrać łącznie:

- 15466 recenzji z serwisu filmweb.pl, z czego 9894 posiadało ocenę liczbową;
- 3121 recenzji z serwisu filmaster.com, z czego 1984 posiadało ocenę liczbową;
- 425 recenzji z serwisu filmawka.pl, z czego 358 posiadało ocenę liczbową;

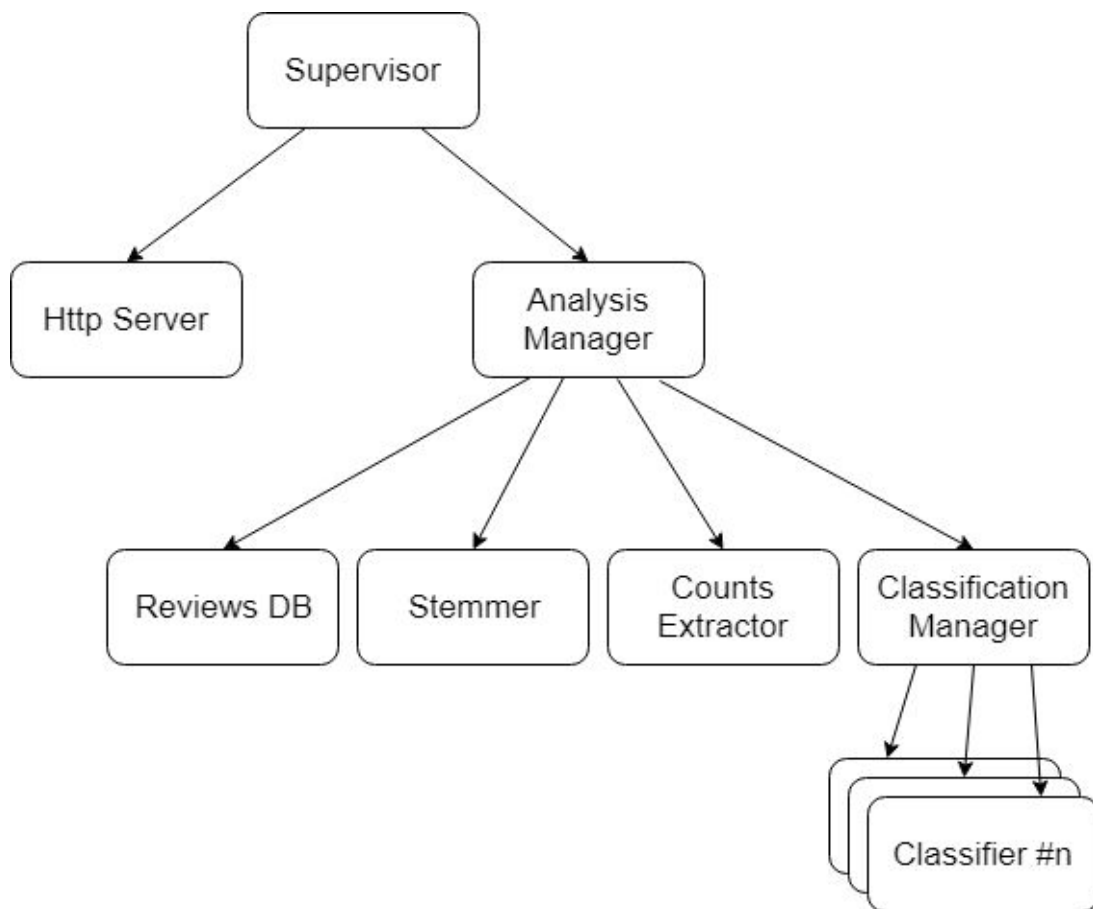
W sumie udało się pobrać 19012 recenzji z czego 12236 posiadało ocenę liczbową. Rozkład ocen wyglądał następująco:

Ocena	Ilość recenzji
10.0	1100
9.0	1489
8.0	2528
7.0	2652
6.0	1828
5.0	1120
4.0	646
3.0	509
2.0	221
1.0	142

W tabeli widać, że niemal 70% recenzji mieści się w przedziale ocen 6.0 - 9.0. Przewidujemy, że z tego powodu jakość oceniania recenzji przypadającej w tym przedziale będzie dużo lepsza, niż poza nią.

III. Architektura

Zgodnie z założeniami, system został zrealizowany w oparciu o techniki aktorowe. Starano się przy tym możliwie jak najbardziej zmniejszyć zależności między aktorami, co w efekcie dało stosunkowo prostą architekturę. Hierarchia aktorów prezentuje się następująco:



Aktorem nadrzędnym jest `Supervisor`. Odpowiada on za inicjalizowanie aktorów `HttpServer` oraz `AnalysisManager`. Odpowiada także za odzyskiwanie aktorów, które ulegną awarii.

`HttpServer` to aktor odpowiadający za przekazywanie zapytań HTTP do pozostałych agentów. Aktualnie zaimplementowano w nim tylko jeden endpoint - GET `analyze`, który służy do przesyłania recenzji w postaci tekstowej do analizatora

`AnalysisManager` spina w całość cały system analizy recenzji filmowych. Odpowiada za przesyłanie składowych wyników do poszczególnych aktorów podrzędnych. Komunikacja z nim przebiega poprzez zapytania `InitializeRequest` oraz `AnalyzeTextRequest`.

`ReviewsDB` zarządza bazą danych z recenzjami. Aktualnie ma ona postać pliku w formacie CSV, składającego się zasadniczo z dwóch kolumn: etykiety (oceny) oraz tekstu (recenzji). Pobieranie recenzji wykonuje się z użyciem zapytania `GetReviewsRequest`.

`Stemmer` odpowiada za preprocessing recenzji. Żądania do niego kierowane są z użyciem zapytania `StemmingsRequests`. Na wejściu otrzymuje zestaw surowych tekstów, a na wyjściu produkuje ich taką samą ilość, tyle że przetworzonych.

`CountsExtractor` odpowiada za ekstrakcję cech z recenzji metodą *Bag of Words*. Komunikacja z nim przebiega z użyciem komunikatów `AddTextsRequests` oraz `ExtractFeatures`. Pierwszy z nich odpowiada za budowanie modelu BOW, zaś drugi przetwarza zestaw tekstów na ich wektorowe odpowiedniki.

`ClassifierManager` pełni rolę koordynatora w procesie klasyfikacji tekstów. Odpowiada za rozsyłanie zapytań między klasyfikatory, zbieranie od nich odpowiedzi i podejmowanie decyzji co do ostatecznej oceny. Bierze również czynny udział w procesie trenowania klasyfikatorów. Interfejs komunikacyjny składa się z dwóch metod: `TrainRequest` oraz `CalculateMarks`.

W ramach systemu zaprojektowano wiele typów klasyfikatorów. Jako że wszystkie z nich dostępne są w bibliotece *Spark*, proces implementacji klasyfikatorów sprowadził się do zaimplementowania aktorów bazowych (`BinaryClassifier` oraz `MultilabelClassifier`) oraz transformacji danych wejściowych. W systemie zaimplementowano następujące klasyfikatory:

- `(Binary/Multilabel)NaiveBayesClassifier` - klasyfikacja techniką naiwnego Bayesa, w wersjach odpowiednio binarnej i wieloklasowej
- `SVMClassifier` - klasyfikacja techniką maszyny wektorów nośnych (ang. *Support Vector Machine*)
- `MLPClassifier` - perceptron wielowarstwowy
- `DTClassifier` - klasyfikacja z użyciem drzewa decyzyjnego
- `OVRClassifier` - klasyfikacja komitetem klasyfikatorów liniowych

IV. Opis działania systemu

Korzystanie z systemu odbywa się zasadniczo w dwóch krokach. Pierw należy wykonać jego inicjalizację, a następnie można przeprowadzać klasyfikację recenzji. Inicjalizacja jest krokiem obowiązkowym ze względu na potrzebę trenowania klasyfikatorów. Dopóki się ona nie wykona, serwer HTTP nie zostanie uruchomiony i zapytania nie będą przekazywane.

Procedura inicjalizacji składa się z następujących kroków:

1. Po uruchomieniu programu `Supervisor` wysyła `InitializeRequest` do `AnalysisManager-a`.
2. `AnalysisManager` wykonuje interakcję ze swoimi aktorami w celu wytrenowania klasyfikatorów. Na początku wysyła `GetReviewsRequest` do `ReviewsDB`.
3. `ReviewsDB` wczytuje dane trenujące z pliku CSV i zwraca je do `AnalysisManager`.
4. `AnalysisManager` wysyła wczytane dane w celu przetworzenia kolejno do `Stemmer-a` (`StemmingsRequest`) i `CountsExtractor-a` (`ExtractFeaturesRequest`)
5. Przetworzone dane przesyłane są do `ClassificationManager-a`, który wysyła je dalej do klasyfikatorów (`TrainRequest`).
6. Po zakończeniu procedury trenowania `AnalysisManager` odsyła do `Supervisor` wiadomość `InitializeResponse`.
7. `Supervisor`, po otrzymaniu potwierdzenia inicjalizacji, wysyła wiadomość `StartServer` do `HttpServer-a`, z żądaniem uruchomienia serwera HTTP.

Po udanej inicjalizacji system jest w stanie przeprowadzać klasyfikację recenzji.

Obliczanie ocen wykonuje się w następujący sposób:

1. Aby otrzymać ocenę recenzji należy wysłać żądanie HTTP GET z treścią recenzji w ciele zapytania,
2. Po otrzymaniu żądania `HttpServer` parsuje wiadomość i wysyła `AnalyseTextRequest` do `AnalysisManager-a`.
3. `AnalysisManager` wysyła `StemmingsRequest` do preprocessora tekstu, a otrzymany wynik przekazuje do `CountsExtractor-a` (`ExtractFeaturesRequest`).

Tak przetworzona recenzja wysyłana jest do `ClassificationManager-a` (`CalculateMarkRequest`).

4. `ClassificationManager` wysyła `CalculateMarkRequest` do wszystkich klasyfikatorów, następnie zbiera odpowiedzi i oblicza ostateczny wynik, który równy jest średniej z najczęściej występujących ocen.

W systemie zaimplementowano ponadto obsługę podstawowych błędów w komunikacji międzyaktorowej. Z wykorzystaniem `Supervisor-a` udało się przechwycić sytuację, gdy któryś z aktorów przestanie działać. System wtedy działa dalej, a wyniki obliczane są przez pozostałe aktory. W celach testowych zaimplementowano mechanizm do symulowania takiej sytuacji. Do serwera HTTP został dodany endpoint z metodą `DELETE`, który przyjmuje parametr `{actorId}`. Po otrzymaniu tego żądania następuje wysłanie wiadomości `Kill` do aktora o identyfikatorze `{actorId}`.

V. Przetwarzanie tekstu

Recenzje załadowane do systemu muszą być przetwarzane przed użyciem ich w klasyfikatorach. Wykonywane jest to poprzez aktory *Stemmer* oraz *Counts Extractor*. Aktor *Stemmer* pełni tutaj trzy funkcje. Po pierwsze tokenizuje otrzymany tekst do postaci pojedynczych słów. Osiągnięto to przy pomocy odpowiedniego wyrażenia regularnego:

`\b\p{L}+\b`

Wyrażenie to znajduje wszystkie ciągi znaków alfanumerycznych włączając to znaki diakrytyczne, oddzielone znakami, będącymi granicą słowa, czyli wszelkiego rodzaju znakami białymi, lub znakami przestankowymi. Dla tak otrzymanych tokenów przeprowadzany jest stemming. Wykorzystany został w tym celu stemmer języka polskiego dostępny na stronie:

<http://morfologik.blogspot.com>

Problematyczne były słowa, dla których stemmer nie był w stanie wygenerować odwzorowania. Problem ten można rozwiązać na dwa sposoby, poprzez pozostawienie słowa w formie niezminionej lub usunięcie tokenu w całości. W trakcie badań przetestowaliśmy oba podejścia.

Ostatnim krokiem przetwarzania tekstu jest odrzucenie tzw. *stopwords*, czyli słów nie niosących dużej wartości znaczeniowej dla zdania. Lista słów została pobrana z repozytorium dostępnego pod linkiem:

<https://github.com/bieli/stopwords>

Następnie, otrzymana lista tokenów przekazywana jest do aktora `CountsExtractor`, który przetwarza ją na model *bag of words*, czyli wektorów liczbowych określających wystąpienia słów modelowych w każdym z tekstów. Pod spodem korzysta on z, dostarczonego wraz z biblioteką Spark, modułu `CountVectorizer`. Otrzymane wektory liczbowe stanowią podstawę dla klasyfikatorów, które wykorzystują je potem do trenowania i predykcji oceny.

VI. Testowanie

Testowanie skuteczności systemu odbyło się w sposób analogiczny do testowania pojedynczego klasyfikatora. Dane zostały podzielone na zbiór uczący oraz zbiór testowy w stosunku 5:1. Zbiór uczący wykorzystano do trenowania klasyfikatorów, a następnie zapytano system o ocenę każdej z recenzji ze zbioru testowego. Taki test przeprowadzono dla różnych konfiguracji klasyfikatorów. Do porównania wykorzystano następujące metryki:

A. Dokładność

$$A = \frac{Cp}{N}$$

B. Średni błąd

$$E = \frac{1}{N} \sum_{j=1}^N |y_j - \hat{y}_j|$$

C. Największy błąd

$$E_{max} = \max(|y_j - \hat{y}_j|)$$

Gdzie:

Cp - liczba poprawnie zaklasyfikowanych recenzji,

N - liczba wszystkich recenzji testowych,

y_j - prawdziwa ocena liczbową j-tej recenzji,

\hat{y}_j - ocena liczbową j-tej recenzji zwrócona przez system

Szczegółowe wyniki badań zaprezentowano w poniższej tabeli:

Konfiguracja klasyfikatorów	Dokładność	Średni błąd	Maksymalny błąd
10 klasyfikatorów Naive Bayes z parametrem smoothing = 1.875	0.238	1.450	9.0
10 klasyfikatorów Naive Bayes z parametrem smoothing = 0.1	0.241	1.439	9.0
10 klasyfikatorów Naive Bayes z parametrem smoothing = 0.1 (oraz usuwaniem tokenów niestemowalnych)	0.234	1.348	9.0
10 klasyfikatorów Logistic Regression w strategii One vs Rest	0.165	1.787	9.0
3 klasyfikatory Naive Bayes 3 klasyfikatory Decision Tree 4 klasyfikatory Logistic Regression w strategii One vs Rest	0.772	0.358	7.0
10 klasyfikatorów MLP w konfiguracji neuronów [5, 10]	0.247	1.303	9.0

Potwierdziła się również hipoteza, iż jakość oceny będzie dużo lepsza dla recenzji o ocenach w skali 6.0 - 9.0. Sprawdzone ją licząc powyższe metryki dla recenzji każdej klasy z osobna. Wykorzystane zostały do tego wyniki z konfiguracji: 10 klasyfikatorów Bayesowskich z parametrem smoothing = 1.875.

Ocena	Dokładność	Średni błąd	Maksymalny błąd
10.0	0.323	1.315	7.0

9.0	0.188	1.071	8.0
8.0	0.378	0.879	7.0
7.0	0.305	0.998	6.0
6.0	0.190	1.407	5.0
5.0	0.091	1.989	5.0
4.0	0.028	2.616	6.0
3.0	0.050	3.590	7.0
2.0	0.000	4.056	7.0
1.0	0.000	5.517	9.0

Jak widać, hipoteza ta okazała się prawdziwa. Mimo, iż dokładność klasyfikacji nie była wysoka, to w zakresie ocen 7.0 - 9.0, gdzie było najwięcej recenzji, klasyfikator średnio mylił się o nie więcej niż 1 w ocenie. W przypadku oceny 5.0 i niższych średni błąd wynosił co najmniej ~2, a w najgorszym przypadku - oceny 1.0, gdzie recenzji było najmniej, klasyfikator nie trafił z żadną oceną i mylił się średnio o 5.5 oceny.

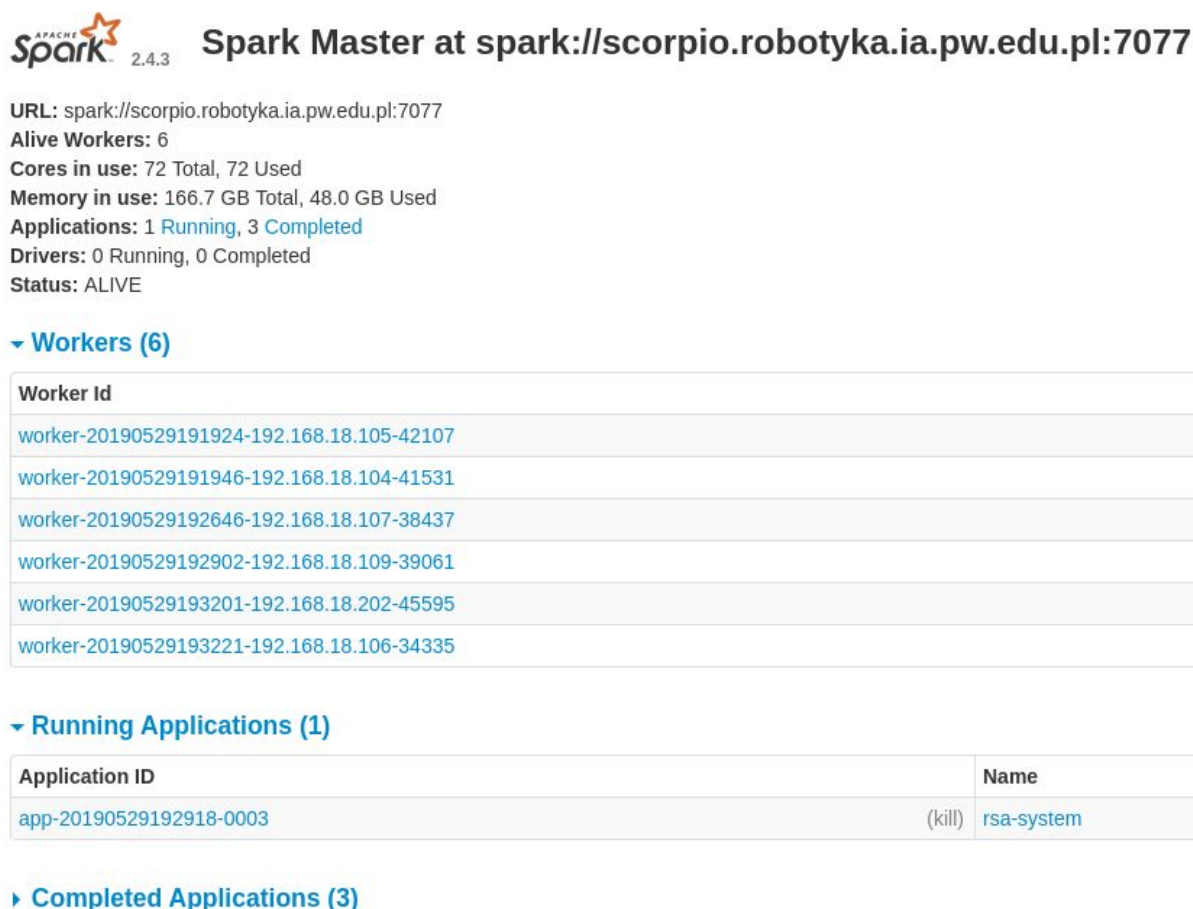
VII. Obliczenia rozproszone

W związku z wykorzystaniem biblioteki Spark do realizowania zadań uczenia maszynowego, postanowiono wypróbować wbudowane mechanizmy do prowadzenia obliczeń w sposób rozproszony. Przetestowano zasadniczo dwie konfiguracje działania:

1. Pojedyncza maszyna wieloprocessorowa i wielordzeniowa,
2. Wiele maszyn wielordzeniowych

Zaskakująco prosto przebiegła konfiguracja *Spark*-a do pracy w środowisku rozproszonym. Wystarczyło tylko w odpowiednim pliku konfiguracyjnym wyspecyfikować nazwy maszyn typu *slave*, uruchomić węzeł typu *master*, podmienić adres węzła typu *master* w pliku

wykonanym i obliczenia natychmiast przenoszone były na sąsiednie komputery. Poniżej zamieszczono zrzut ekranu z interfejsu kontrolnego Spark-a:



The screenshot displays the Spark Master web interface. At the top, it shows the Apache Spark logo (version 2.4.3) and the title "Spark Master at spark://scorpio.robotyka.ia.pw.edu.pl:7077". Below this, system statistics are listed: URL, Alive Workers (6), Cores in use (72 Total, 72 Used), Memory in use (166.7 GB Total, 48.0 GB Used), Applications (1 Running, 3 Completed), Drivers (0 Running, 0 Completed), and Status (ALIVE). A section titled "Workers (6)" contains a table listing six worker nodes with their IDs and IP addresses. Another section titled "Running Applications (1)" contains a table with one entry for an application named "rsa-system". A third section titled "Completed Applications (3)" is partially visible at the bottom.

Spark Master at spark://scorpio.robotyka.ia.pw.edu.pl:7077

URL: spark://scorpio.robotyka.ia.pw.edu.pl:7077
Alive Workers: 6
Cores in use: 72 Total, 72 Used
Memory in use: 166.7 GB Total, 48.0 GB Used
Applications: 1 Running, 3 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

▼ Workers (6)

Worker Id
worker-20190529191924-192.168.18.105-42107
worker-20190529191946-192.168.18.104-41531
worker-20190529192646-192.168.18.107-38437
worker-20190529192902-192.168.18.109-39061
worker-20190529193201-192.168.18.202-45595
worker-20190529193221-192.168.18.106-34335

▼ Running Applications (1)

Application ID	Name
app-20190529192918-0003	(kill) rsa-system

► Completed Applications (3)

VIII. Podsumowanie

Założeniem projektu było stworzenie systemu agentowego, potrafiącego odzwierciedlić recenzję napisaną językiem naturalnym, w postaci oceny liczbowej. System ten udało się zrealizować, z dokładnością do przyjętych założeń. W oparciu o dane uczące pozyskane przez nas, z przeprowadzonych badań wynika, iż system ten jest w stanie klasyfikować recenzje filmowe z dokładnością sięgającą nawet 77%. W projekcie wykorzystano szereg różnych metod klasyfikacji, przy czym najlepsze rezultaty uzyskaliśmy poprzez zmieszanie zarówno drzew decyzyjnych, klasyfikatorów OVR oraz typu Naiwny Bayes. Wykorzystanie bibliotek Akka i Spark znacząco usprawniło wykonywanie zadania oraz pozwoliło wykonywać obliczenia w sposób rozproszony.

