Warszawa, 12.06.2018

SAG & WEDT
Projekt- SAG_WEDT_brand_safety

Dokumentacja końcowa

System „brand safety", analizujący kontekst dla promowanych treści on-line

Szymon Borodziuk
Maciej Wiraszka
Michał Ziółkowski

1. Interpretacja tematu

Termin *brand safety* oznacza proces, który ma na celu zapobieganie wyświetlaniu reklamy na stronie, która może być potencjalnie szkodliwa dla marki [1]. Komercyjnie dostępne rozwiązania zarządzania treścią reklamową oferują szereg predefiniowanych filtrów oraz aktywny monitoring docelowych stron w celu zapobiegnięcia prezentacji reklam o treści nieodpowiedniej dla danego typu witryny [2].

Opisywany w tym dokumencie system *SAG_WEDT_Brand_Safety_2018* pozwala określić na podstawie treści tekstowej strony czy reklama powinna się pojawić, czy nie. Do tego celu wykorzystana została implementacja zbioru metod NLP (ang. *natural language processing*) opisanych szerzej w dalszych rozdziałach.

System pozwala na określenie różnych kryteriów determinujących poprawność treści dla prezentowanej reklamy.

2. Rozważane narzędzia i technologie

SAG_WEDT_Brand_Safety_2018 napisany został w języku Java. Projekt oparty jest na bibliotece Akka, umożliwiającej stworzenie skalowalnego, rozproszonego systemu w modelu aktorowym [3].

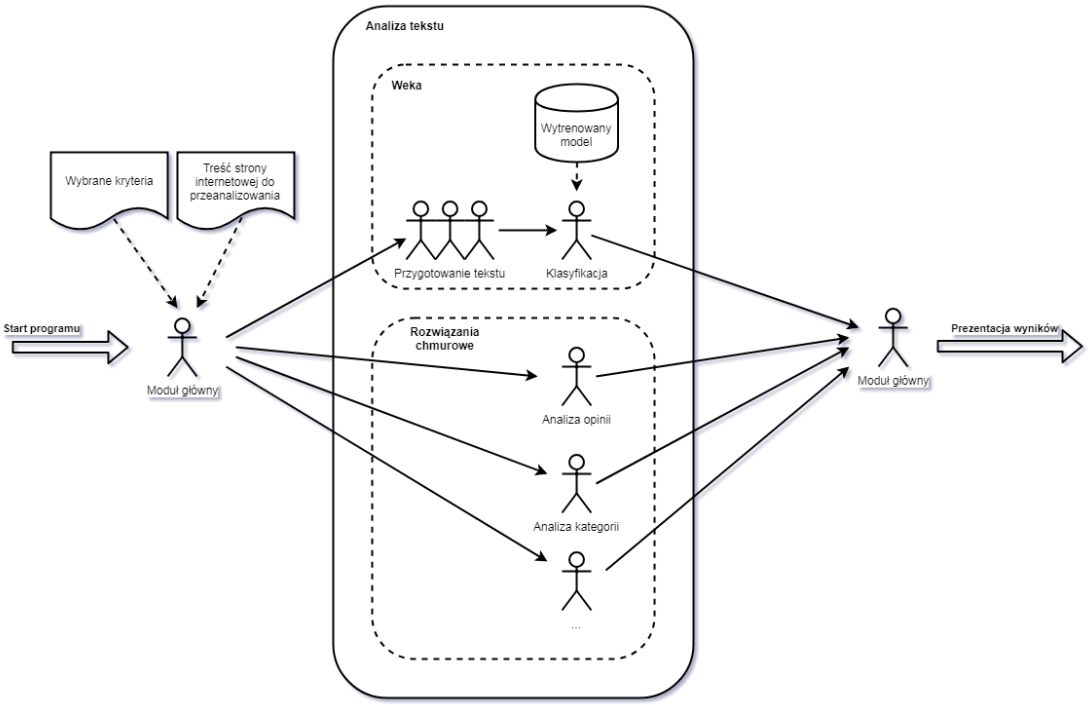
Do klasyfikacji stron zamierzaliśmy wykorzystać dwa podejścia: przetwarzanie chmurowe, wykorzystujące API Google Cloud Engine [4] oraz własną implementację procesu klasyfikacji opartą o biblioteki udostępnione w ramach platformy Weka [5].

Rozwiązanie chmurowe wykorzystywać będzie platformę przetwarzania języka naturalnego udostępnioną w ramach Google Cloud [6]. Wstępnie przetworzony tekst, pobrany z klasyfikowanej strony i oczyszczony przez dedykowanego aktora, zostanie przesłany w zapytaniu RESTowym. Po otrzymaniu odpowiedzi, aktor zwróci wynik zapytania do aplikacji

głównej. Z racji tego, że platforma Google NLP udostępnia kilka interfejsów programistycznych o zróżnicowanych funkcjonalnościach, dla każdej z nich (sentyment, klasyfikacja treści itp.) utworzony zostanie dedykowany aktor.

Do utworzenia i wytrenowania własnego modelu oraz późniejszej klasyfikacji tekstu na stronie wykorzystamy powszechnie uznaną w środowisku naukowym platformę Weka [6][7], zapewniającą szereg bibliotek użytecznych w przygotowaniu zbiorów wejściowych (stemming, lemantyzacja, odsiewanie 'stopwords'), trenowania oraz klasyfikacji tekstu. W tym wypadku odrębni aktorzy zajmować się będą parsowaniem tekstu strony oraz klasyfikacją docelowej treści. Na koniec, wynik klasyfikacji zwrócony zostanie do aplikacji głównej.

3. Zakładana architektura rozwiązania



Użytkownik po uruchomieniu programu specyfikuje kryteria, wg których przeprowadzona zostanie analiza wybranej treści strony. Następnie poszczególne moduły (aktorzy) przetwarzają i analizują współbieżnie tekst w oparciu o przypisane im kryteria. Po zakończeniu pracy poszczególne jednostki informują aktora głównego o wynikach analizy. Aktor główny zbiera wyniki oraz prezentuje je użytkownikowi oraz przedstawia rekomendację wyświetlenia reklamy na danej stronie.

Przykładowe kryteria:

- Opinia (ang. Sentiment) - specyfikuje, czy treść ma pozytywny wydźwięk
- Kategoria (ang. Category) - określa ogólne znaczenie danego tekstu i przypisuje go do danej kategorii

Pojedyncze jednostki-aktorzy realizują określone zadania analizy, odpowiadające kryteriom wyspecyfikowanym przez użytkownika. Część aktorów może być połączona w tzw. *NLP pipeline*, łączący wyjścia i wejścia kolejnych aktorów w sekwencji, odpowiadające kolejnym elementom przetwarzania tekstu.

4. Główne założenia projektowe

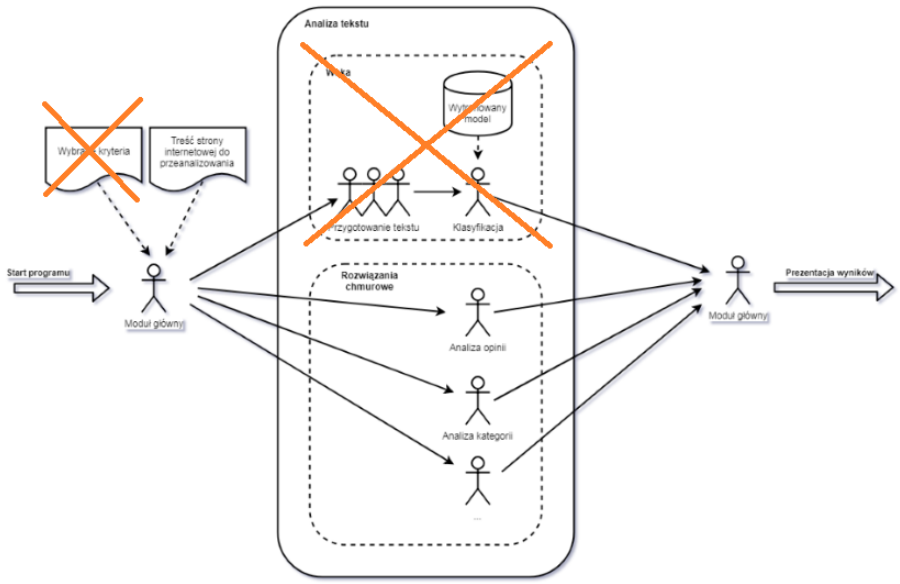
- Analiza treści strony będzie przeprowadzana na wyekstrahowanych danych tekstowych ze strony, co zostanie przeprowadzone poza systemem *SAG_WEDT_Brand_Safety_2018*. Na potrzeby testowania projektu dobierzemy treść zbioru stron o różnym charakterze.
- System będzie odporny na awarię / wyłączenie części jednostek (aktorów do analizy tekstu). W przypadku awarii części aktorów analizujących tekst, ocena poprawności treści strony zostanie zaprezentowana użytkownikowi (w oparciu o wyniki pozostałych jednostek analizujących).
- Na ten moment projekt zakłada dwie formy klasyfikacji tekstu (chmurowa + własny klasyfikator), jednakże rozpatrujemy rozszerzenie lub zawężenie powyższych sposobów analizy w zależności od osiąganych efektów.

5. Zaimplementowane rozwiązanie

5.1. Różnice względem założeń

- Do analizy języka naturalnego wykorzystujemy wyłącznie platformę przetwarzania języka naturalnego udostępnioną w ramach Google Cloud Engine.
- Kryteria, według których przeprowadzana jest analiza treści strony zaimplementowane są w ciele poszczególnych aktorów - klasyfikatorów.

5.2. Architektura rozwiązania



5.3. Aktorzy – opis systemu

System aktorów został stworzony przy użyciu biblioteki Akka wraz z modułem Akka – cluster. Akka – cluster pomaga w stworzeniu abstrakcji systemu aktorów działających na wielu maszynach (u nas symulowane przez włączeniu aplikacji na wielu portach). Umożliwia również ustawienie podstawowych kwestii dotyczących działania klastra takie jak ilość root node'ów oraz minimalną ilość aktorów potrzebną do wystartowania klastra poprzez zdefiniowanie pliku konfiguracyjnego (*"aplication.conf"*). Klaster monitoruje również aktywność aktorów w systemie i w przypadku błędu aktora wysyła do monitorujących aktorów wiadomość o błędzie danego aktora.

Architektura systemu została stworzona w taki sposób, że możliwe jest dodawanie dowolnej ilości aktorów tego samego typu. Aktorzy klasyfikujący po dołączeniu do systemu (klastra) wysyłają zgłoszenie do aktorów frontendowych o swojej aktywności. Aktorzy frontendowi na podstawie zgłoszeń aktorów klasyfikujących tworzą listy aktorów dostępnych dla każdego typu klasyfikacji.

Architektura systemu:

- Klaster umożliwia dodawanie równoległych aktorów do systemu w dowolnej ilości dzięki czemu system może działać w wielu różnych konfiguracjach ilości aktorów klasyfikujących i frontendowych
- Aktorzy klasyfikujący po wystartowaniu i dołączeniu do systemu wysyłają do aktorów frontendowych wiadomość o swojej dostępności. Na podstawie tych widomości aktorzy frontedowi tworzą listy dostępnych aktorów danego typu klasyfikatora, do których wysyłają zapytania o klasyfikacje strony.
- W przypadku błędu aktora klasyfikującego, klaster wysyła do aktorów frontendowych wiadomość o braku dostępności danego aktora. Aktor frontendowy usuwa go z listy dostępnych klasyfikatorów.
- Klient wysyła wiadomość do aktora frontendowego a on na podstawie typu wiadomości przesyła ją do danego aktora klasyfikującego lub do wszystkich aktorów klasyfikujących. Aktor frontendowy wybiera kolejno typ aktora z listy i wysyła do niego zapytanie. Dzięki temu można podzielić obciążenie klasyfikacji na wiele aktorów (docelowo maszyn).
- Każdy aktor klasyfikujący tworzy dla każdego zapytania nowego potomka, który przetwarza otrzymany kontent strony i wysyła odpowiedź do aktora frontentowego. Dzięki temu każde zapytanie działa niezależnie i nie musi czekać na wykonanie zapytania przez aktora głównego.
- W przypadku braku aktora klasyfikującego aktor frontendowy próbuje określoną ilość razy wysłać zapytanie ponowne do danego typu aktora klasyfikującego. Po wyczerpaniu puli zapytań zwraca do klienta wiadomość o braku danego aktora.
- Liczba ponownych zapytań i czasu interwału sprawdzania czy aktor jest dostępny jest ustalany w pliku frontend.conf

5.4. Aktorzy - opis aktorów

- Aktor główny (frontendowy)

Aktor odpowiedzialny za pobieranie treści stron od klienta oraz wysłanie ich do odpowiedniego aktora klasyfikującego. Odpowiada również za pobranie i odesłanie odpowiedzi do klienta. Obsługuje zarówno odpowiedzi statusowe aktorów (otrzymane od aktorów klasyfikujących) jak również odpowiedzi o błędach systemu lub braku działania różnych typów aktorów klasyfikacji.

- Aktor klasyfikujący sentyment (opinie)

Aktor wykorzystuje REST API Sentiment Analysis udostępnione w ramach Google Cloud Engine NLP. Po otrzymaniu od aktora głównego przeprocesowanego tekstu klasyfikowanej strony, aktor przesyła tekst w formie zapytania HTTP na platformę Google NLP. Tam dokonywana jest właściwa analiza sentymentu tekstu. Z odpowiedzi platformy ekstrahowane są dwa parametry: średni sentyment klasyfikowanej treści oraz ważność klasyfikacji. W zależności od ich wartości aktor zwraca jedną z trzech wartości nominalnych określających sentyment wiadomości: pozytywną, negatywną lub też nieokreśloną.

- Aktor klasyfikujący przynależność do kategorii

Aktor wykorzystuje REST API Content Classification udostępnione w ramach Google Cloud Engine NLP. Po otrzymaniu od aktora głównego przeprocesowanego tekstu klasyfikowanej strony, aktor przesyła tekst w formie zapytania HTTP na platformę Google NLP. Tam dokonywana jest klasyfikacja strony oraz przydział do predefiniowanych kategorii treściowych. Z odpowiedzi platformy ekstrahowana jest lista klucz-wartość w postaci kategorii i procentowego prawdopodobieństwa, że treści jej odpowiadające znajdowały się w klasyfikowanych tekście. W zależności od listy kategorii zastrzeżonych podanych w konfiguracji aktora, na podstawie odpowiedzi Google Cloud, aktor zwraca jedną z trzech wartości nominalnych określających kontent strony: pozytywny, negatywny lub też nieokreślony.

5.5. Link do repozytorium kodu:

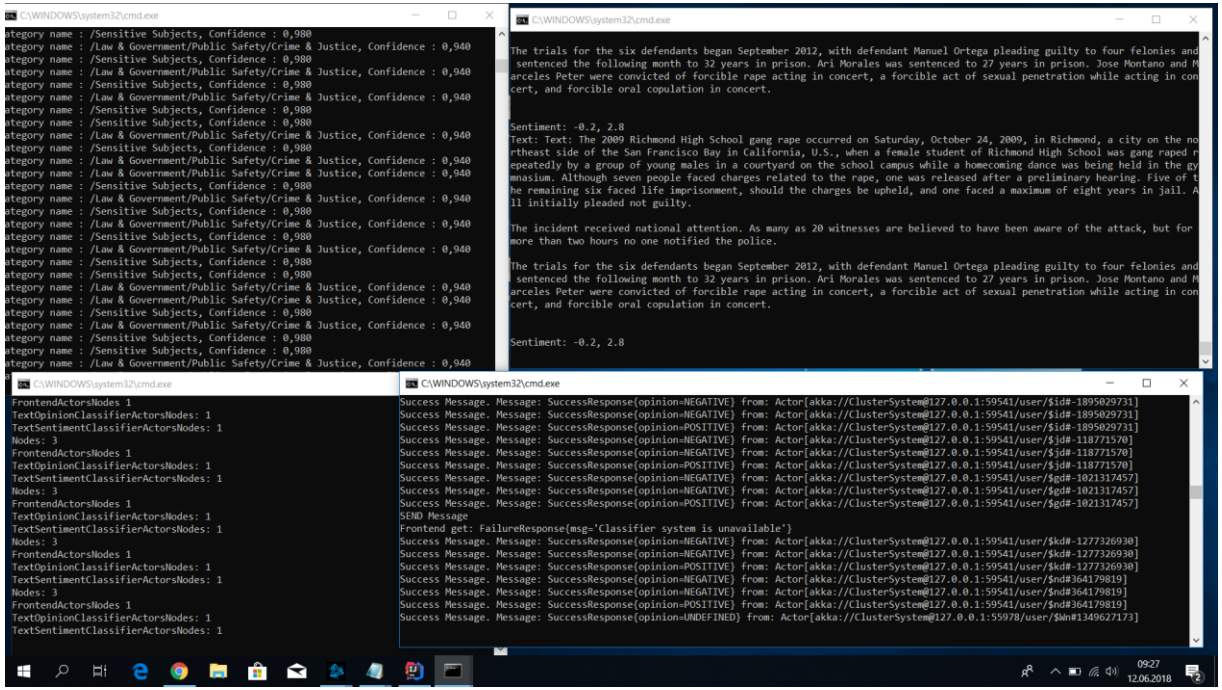
https://github.com/Zwirek009/SAG_WEDT_brand_safety

6. Testy rozwiązania

6.1. Z włączonymi aktorami – klasyfikatorami

Na poniższym zrzucie ekranu prezentowana jest prawidłowa praca systemu z włączonymi wszystkimi aktorami klasyfikującymi. Widać na nim jak przetwarzana jest wiadomość przez każdego aktora klasyfikującego oraz wiadomości wysyłane i otrzymywane przez aktora głównego.

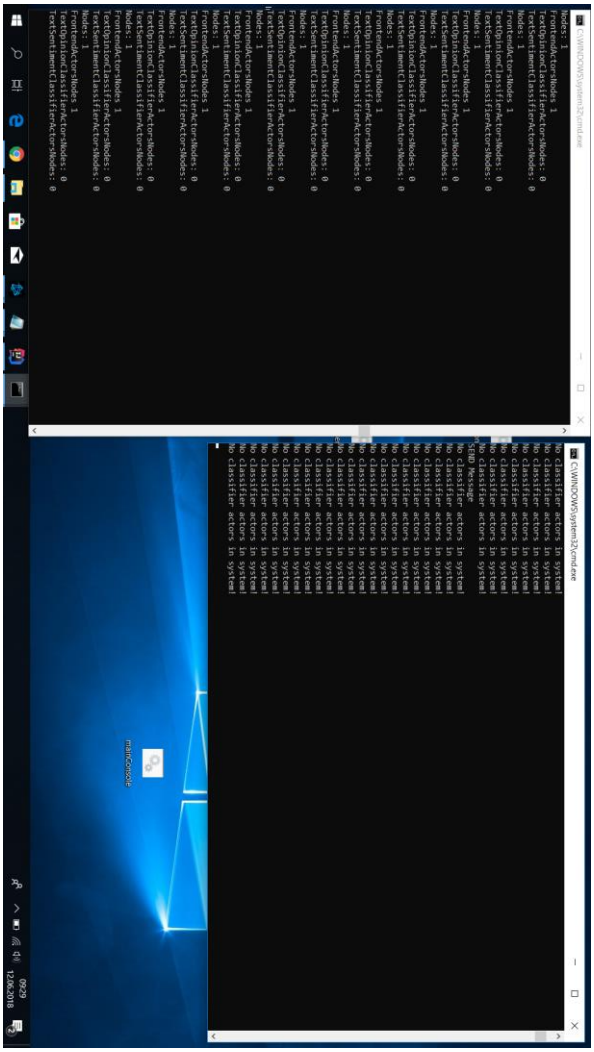
(zrzut ekranu w lepszej jakości: https://github.com/Zwirek009/SAG_WEDT_brand_safety/blob/master/wlaczone_klasyfikatory.png)


C:\WINDOWS\system32\cmd.exe
ategory name : /Sensitive Subjects, Confidence : 0,980
ategory name : /Law & Government/Public Safety/Crime & Justice, Confidence : 0,940
The trials for the six defendants began September 2012, with defendant Manuel Ortega pleading guilty to four felonies and
sentenced the following month to 32 years in prison. Ari Morales was sentenced to 27 years in prison. Jose Montano and M
arceles Peter were convicted of forcible rape acting in concert, a forcible act of sexual penetration while acting in con
cert, and forcible oral copulation in concert.
Sentiment: -0.2, 2.8
Text: Text: The 2009 Richmond High School gang rape occurred on Saturday, October 24, 2009, in Richmond, a city on the no
rtheast side of the San Francisco Bay in California, U.S., when a female student of Richmond High School was gang raped r
epeatedly by a group of young males in a courtyard on the school campus while a homecoming dance was being held in the gy
mnasium. Although seven people faced charges related to the rape, one was released after a preliminary hearing. Five of t
he remaining six faced life imprisonment, should the charges be upheld, and one faced a maximum of eight years in jail. A
ll initially pleaded not guilty.
The incident received national attention. As many as 20 witnesses are believed to have been aware of the attack, but for
more than two hours no one notified the police.
FrontendActorsNodes 1
TextOpinionClassifierActorsNodes: 1
TextSentimentClassifierActorsNodes: 1
Nodes: 3
SEND Message
Frontend get: FailureResponse{msg='Classifier system is unavailable'}
Success Message. Message: SuccessResponse{opinion=NEGATIVE} from: Actor[akka://ClusterSystem@127.0.0.1:59541/user/$id#-1895029731]
Success Message. Message: SuccessResponse{opinion=POSITIVE} from: Actor[akka://ClusterSystem@127.0.0.1:59541/user/$id#-1895029731]
Success Message. Message: SuccessResponse{opinion=NEGATIVE} from: Actor[akka://ClusterSystem@127.0.0.1:59541/user/$jd#-118771570]
Success Message. Message: SuccessResponse{opinion=POSITIVE} from: Actor[akka://ClusterSystem@127.0.0.1:59541/user/$jd#-118771570]
Success Message. Message: SuccessResponse{opinion=NEGATIVE} from: Actor[akka://ClusterSystem@127.0.0.1:59541/user/$gd#-1021317457]
Success Message. Message: SuccessResponse{opinion=POSITIVE} from: Actor[akka://ClusterSystem@127.0.0.1:59541/user/$gd#-1021317457]
Success Message. Message: SuccessResponse{opinion=NEGATIVE} from: Actor[akka://ClusterSystem@127.0.0.1:59541/user/$kd#-1277326930]
Success Message. Message: SuccessResponse{opinion=POSITIVE} from: Actor[akka://ClusterSystem@127.0.0.1:59541/user/$kd#-1277326930]
Success Message. Message: SuccessResponse{opinion=NEGATIVE} from: Actor[akka://ClusterSystem@127.0.0.1:59541/user/$nd#364179819]
Success Message. Message: SuccessResponse{opinion=POSITIVE} from: Actor[akka://ClusterSystem@127.0.0.1:59541/user/$nd#364179819]
Success Message. Message: SuccessResponse{opinion=UNDEFINED} from: Actor[akka://ClusterSystem@127.0.0.1:55978/user/$kn#1349627173]
09:27
12.06.2018

6.2. Z wyłączonymi aktorami - klasyfikatorami

Poniższy system ukazuje jak aktor frontendowy reaguje na brak aktorów klasyfikujących. Komunikuje o braku aktorów klasyfikujących oraz pokazuje wiadomość o błędzie wysyłaną do klienta po wyczerpaniu ilości dostępnych odpowiedzi dla danej odpowiedzi.

(zrzut ekranu w lepszej jakości: https://github.com/Zwirek009/SAG_WEDT_brand_safety/blob/master/wylaczone_klasyfikatory.png)



7. Bibliografia

1. Definicja "brand safety" - https://programmatic.pl/slownik-rtb/
2. Filtry AdSense - https://support.google.com/adsense/answer/2369326?hl=en
3. Witryna projektu AKKA - https://akka.io/
4. Witryna GCE - https://cloud.google.com/compute/
5. Witryna GC NLP - https://cloud.google.com/natural-language/
6. Witryna projektu WEKA - https://www.cs.waikato.ac.nz/ml/weka/
7. Ranking popularności platform data mining'owych - https://www.kdnuggets.com/2014/06/kdnuggets-annual-software-poll-rapidminer-continues-lead.html