Dokumentacja końcowa projektu
Projekt łączony z przedmiotów *Systemy Agentowe* oraz *Wstęp do eksploracji danych tekstowych w sieci WWW*

Andrzej Dawidziuk, Tomasz Kogowski, Rafał Okuniewski

22 stycznia 2018

1 Wstęp

1.1 Zadanie projektowe

Zadanie projektowe polega na wyszukiwaniu tematycznych wiadomości w Internecie oraz ich analizie semantycznej pod kątem podobieństwa i tematyki. Na potrzeby projektu analizowane będą portale informacyjne udostępniające treści w języku angielskim.

Celem projektu było stworzenie rozproszonego systemu agentowego, który umożliwi zapisywanie oraz wektoryzację artykułów z różnych internetowych serwisów informacyjnych, oraz umożliwi wskazanie artykułów o podobnej treści, odnoszących się do podobnych tematów, wydarzeń, wypowiedzi.

1.2 Wykorzystane technologie

W celu realizacji systemu agentowego wykorzystano powszechnie stosowany framework Akka[1]. Umożliwia on stworzenie funkcjonalnego systemu agentowego w dwóch językach - Scala oraz Java. Zespół podjął decyzję o implementacji w języku Scala.

Warstwę bazodanową, w której przechowywane były informacje o pobranych ze stron informacyjnych artykułach, stanowiła baza danych Postgres.

W celu stworzenia modułu odpowiedzialnego za wektoryzację oraz analizę podobieństwa artykułów wykorzystano język Python oraz biblioteki NLTK i GenSim.

Ze względu na konieczność integracji modułu agentowego z modułem analitycznym, zdecydowano o integracji poprzez wykonywanie zapytań do serwera HTTP w którym osadzony był moduł analityczny. Serwer ten został wykonany z wykorzystaniem biblioteki flask.

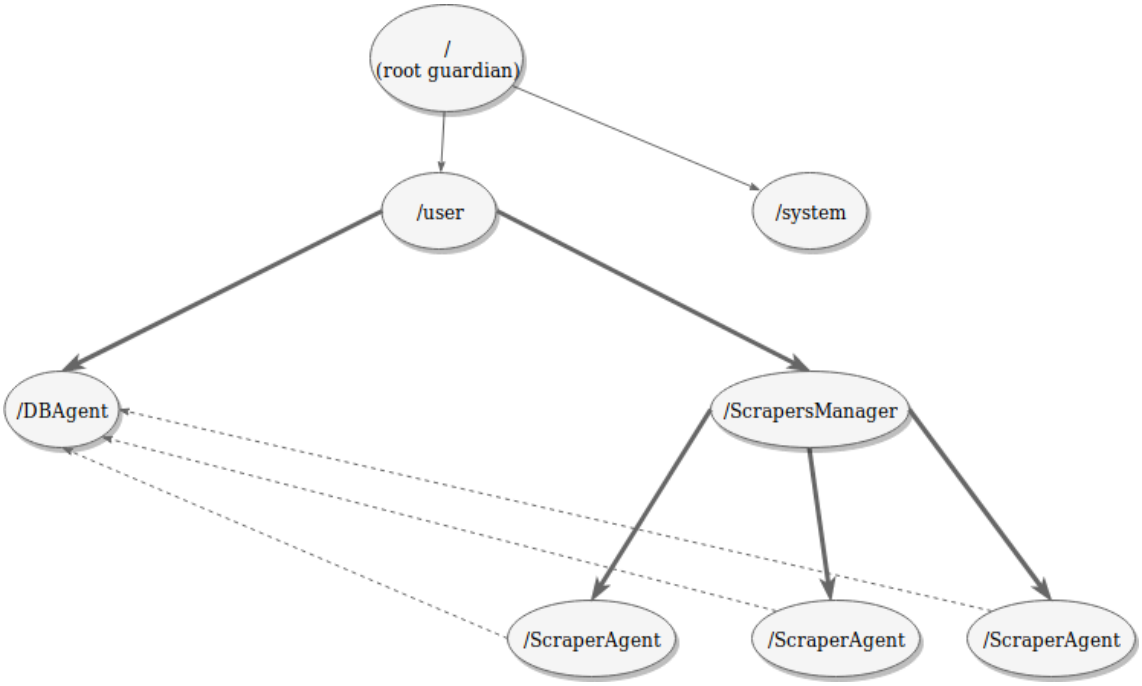
2 System agentowy

Na rys. 1 został umieszczony schemat stworzonej przez zespół architektury systemu agentowego.

Przy czym strzałki pogrubione oznaczają podległość agentów (aktorzy przy końcu strzałki są ”dziećmi” aktorów z początku strzałki), natomiast dodatkowe strzałki przerywane oznaczają kierunek wysyłania wiadomości do agentów.

Poszczególni agenci w systemie różnią się między sobą zadaniami:

- `DBAgent` - pełni zadania agenta mającego bezpośredni dostęp do bazy danych, w której składowane są artykuły. Obsługuje on dwa typy otrzymywanych wiadomości:
  - **SaveArticles** - wiadomość od ScraperAgenta, zlecająca zapisanie do bazy danych listy artykułów
  - **SaveArticle** - wiadomość od ScraperAgenta, zlecająca zapisanie do bazy danych pojedynczego artykułu.
- `ScrapersManager` - agent pełniący obowiązki menedżera ScraperAgentów. Przechowuje on mapę referencji do nich, tworzy je oraz cyklicznie zleca im przeglądanie artykułów ze stron. Agent ten obsługuje dwa typy wiadomości:



Rysunek 1: Schemat wdrożonego systemu zgodnie z architekturą Akka

1. **OrderScrapping** - wiadomość, którą agent wysyła sam do siebie co pewien określony czas (ustawiony z wykorzystaniem schedulera Akki), a która powoduje wysłanie do każdego zarejestrowanego w nim ScraperAgenta wiadomości typu **Scrap**
2. **CreateScraperAgent** - wiadomość od głównego agenta systemu (**user**), która zleca utworzenie ScraperAgenta mającego odpowiadać, za przeglądanie strony kanału RSS o określonym adresie URL.

- `ScraperAgent` - typ agenta odpowiedzialnego za przeglądanie danego kanału RSS, ekstrakcję nowych artykułów, ich wektoryzację oraz wysłanie ich do `DBAgenta` celem zapisu w bazie danych. Obsługuje on jeden typ wiadomości:

  1. **Scrap** - wiadomość od ScrapersManagera, która zleca przejrzenie kanału RSS, pobranie nowych artykułów, ich wektoryzację i wysłanie do zapisu w bazie danych.

  Agent przechowuje informację o tym, kiedy nastąpił ostatni pobór danych z kanału RSS, za który odpowiada i na tej podstawie filtruje artykuły z kanału tak, aby nie wysyłać do zapisu artykułów które już istnieją w bazie danych. Za pomocą obiektu `ScraperHttpService` agent ten komunikuje się z modułem analitycznym wykorzystującym algorytm doc2vec, dzięki któremu następuje wektoryzacja artykułu. Artykuł w pełnej formie (obejmującej treść, adres pobrania, datę publikacji i postać wektorową) zostaje wysłany wraz z wiadomością **SaveArticle** do `DBAgenta`.

System ten został wykonany z użyciem popularnego dla takich celów frameworku Akka oraz języka Scala.

W stworzonym przez zespół systemie część agentów może działać w innych węzłach niż pozostałe. Dzięki wykorzystaniu biblioteki `Akka Remoting`, zespół przetestował umieszczenie osobno agenta ScrapersManager oraz wszystkich agentów typu ScraperAgent.

2.1 Język Scala

Framework Akka umożliwia implementację systemów agentowych z wykorzystaniem dwóch możliwych języków - Java oraz Scala. Oba bazują na maszynie wirtualnej Javy, natomiast znacznie różnią się od siebie wykorzystywanym paradygmatem programowania - Java jest językiem typowo obiektowym, podczas gdy Scala umożlwia zarówno korzystanie z paradygmatu obiektowego, jak i funkcyjnego. Ze względu na potencjalne korzyści wynikające z użycia jej, zwięzłość kodu oraz

lepsze przystosowanie do założeń systemu obiektowego (takich jak asynchroniczność), zespół, pomimo braku znajomości języka na początku realizacji projektu, zdecydował się na implementację wykorzystującą Scalę.

Język Scala okazał się być dla większości członków zespołu językiem trudnym do opanowania o wysokim wymaganym progu wiedzy wejściowej. Pomimo iż dwóch członków zespołu znało paradygmat programowania funkcyjnego z innych języków programowania takich jak Haskell, rozbudowana składnia Scali była na początku trudna do opanowania.

Końcowy efekt w postaci zaimplementowanego projektu udowodnił jednakże, iż opanowanie tego języka było warte czasu przeznaczonego na nie. Scala umożliwia korzystanie z paradygmatu funkcjonalnego w sposób, który pomaga w implementacji zamiast nastręczać problemów (co zdarza się w przypadku języków czysto funkcjonalnych, takich jak Haskell). Jej wielką zaletą jest również możliwość korzystania z klas Javy, co umożliwiło zespołowi korzystanie z popularnych bibliotek służących do przetwarzania kanałów RSS i parsowania stron internetowych. Mechanizmy typowe dla języków funkcyjnych, takie jak "match expressions", pozwoliły wydatnie zredukować ilość kodu konieczną do implementacji systemu. Cała architektura agentowa zaimplementowana przez zespół przy użyciu Scali zajęła zaledwie 240 linii kodu. Jest on również czytelny i łatwo modyfikowalny.

Scala okazała się jednakże nie być właściwym narzędziem jeśli chodzi o stworzenie systemu ORM (ang. *object-relational mapping*). Biblioteka Slick, którą zespół wykorzystał do tego celu, dostarczyła wielu trudności w implementacji.

2.2 Framework Akka

W celu zrealizowania systemu agentowego zespół zdecydował się wykorzystać framework Akka. Decyzja ta podyktowana była chęcią wykorzystania popularnego w tym zakresie frameworku i stworzenia systemu wykorzystującego wszystkie zalety systemów agentowych - możliwość budowy relatywnie nieskomplikowanych w implementacji systemów, które jednocześnie będą wydajne, łatwo rozszerzalne i rozproszone.

Nauka frameworka na podstawie istniejącej dokumentacji była czasochłonna. Dokumentacja zamieszczona na stronie Akki skupia się na zaprezentowaniu zalet korzystania z systemu aktorowego pomijając jednakże szczegóły implementacyjne - prezentuje jedynie dwa przykłady prostego systemu, opisując go w sposób dość rozwlekły. Bez wykorzystania źródeł zewnętrznych, w tym przykładowych systemów zaimplementowanych z wykorzystaniem tej biblioteki, niemożliwe byłoby na przykład zaimplementowanie systemu rozproszonego z wykorzystaniem biblioteki Akka Remote.

Implementacja systemu agentowego ujawniła jednak wszystkie zalety frameworka Akka. Po zapoznaniu sie w dokładny sposób z dokumentacją była ona szybka i nie pociągnęła za sobą znaczących modyfikacji aż do końca projektu, co należy zawdzięczać również przemyślanej uprzednio strukturze systemu.

Wielką zaletą frameworka okazała się łatwość rozbudowywania i rozpraszania aplikacji. Zmiana aplikacji z wersji lokalnej na rozproszoną wymagała niemal wyłącznie zmiany w plikach konfiguracyjnych oraz sposobie adresowania aktorów.

Również rozszerzenie zakresu analizowanych kanałów RSS okazało się być bezproblemowe. W początkowej wersji systemu analizowane były jedynie kanały RSS o różnej tematyce pochodzące z serwisów informacyjnych BBC oraz CNN. W celu powiększenia zbioru testowego do rozsądnej liczby artykułów, zostały zaimplementowane również parsery takich mediów jak:

- Reuters
- Washington Times
- New York Times
- The Guardian

Umożliwiło to zwiększenie bazy danych artykułów ze 137 pozycji do niemal 500, pociągając za sobą niecałą godzinę pracy.

Możliwość łatwej budowy systemów rozproszonych oraz prostego odwzorowania zaplanowanej architektury w kodzie należą do największych zalet Akki dostrzeżonych przez zespół. Cechy te sprawiają, iż decyzję o implementacji z wykorzystaniem tego frameworka należy określić jako zdecydowanie pozytywną.

Istnieją jednakże również negatywne strony takiej decyzji. Systemy agentowe nie należą do najpopularniejszych, co sprawia, że wsparcie społeczności dla frameworka Akka czasami okazywało się

niewystarczające. Zespół napotkał duże problemy przechodząc od wersji lokalnej projektu do wersji rozproszonej ze względu na niejednoznaczne komunikaty o błędach. Zostały one jednak szybko pokonane i trudności takie nie przeważają zalet wykorzystania zarówno architektury agentowej, jak i konkretnego frameworka w postaci Akki.

2.3 Schemat pobierania wiadomości

Agenty odpowiedzialne za pobieranie oraz parsowanie artykułów, uruchamiane są regularnie. Do pobierania danych z kanału RSS wykorzystano bibliotekę `ROME`, która dla zadanego adresu url pobiera tak zwany `feed`. Następnie za pomocą narzędzia `jsoup` pobierane oraz parsowane są treści każdego artykułu. Następnie treść każdego artykułu poddawany jest wektoryzacji w module NLP. W kolejnym kroku najważniejsze informacje o artykule:

- adres strony z której został pobrany,
- strona z której został pobrany,
- text artykułu,
- data publikacji,
- tagi,
- oraz wektor zwrócony przez moduł NLP,

są zapisywane w bazie danych (rys. 2).

Rysunek 2: Schemat bazy danych

3 Przetwarzanie języka naturalnego

Do określania podobieństwa artykułów użyta została technika doc2vec. Przy jej użyciu możliwe jest wygenerowanie dla każdego z artykułów wektora liczb rzeczywistych, osadzającego cały dokument w przestrzeni wektorowej. Wektory dokumentów podobnych do siebie powinny leżeć w tej przestrzeni blisko siebie.

3.1 Doc2Vec

Nazwa doc2vec[2] (lub paragraph2vec) jest wspólną nazwą używaną do określenia dwóch różnych algorytmów:

- Distributed memory model

- Distributed bag of words

Algorytmy te umożliwiają wyznaczenie wektorowej reprezentacji dokumentu za pomocą *uczenia bez nadzoru.* Dzięki temu możliwe jest wytrenowanie modelu doc2vec, służącego do wyznaczania wektorów dla jeszcze nie widzianych artykułów na nieoznaczonym korpusie.

Doc2vec jest rozszerzeniem techniki word2vec służącej do wyznaczania wektorowych reprezentacji pojedynczych słów na podstawie kontekstu w jakim występują w korpusie źródłowym. Tak więc do jego zastosowania potrzebne są reprezentacje słów (przygotowane wcześniej lub wyznaczone podczas treningu modelu doc2vec). Oczywistym ograniczeniem algorytmu jest to, że lista słów rozpoznawanych przez model ograniczona jest do słów występujących w korpusie treningowym. Tak więc korpus treningowy powinien być jak najbardziej zbliżony do rzeczywistych artykułów.

3.1.1 Podobieństwo wektorów

Podobieństwo wektorów wyznaczonych przez model doc2vec określane jest przez wyznaczenie iloczynu skalarnego ich wersorów:

$$S(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u}}{||\mathbf{u}||} \cdot \frac{\mathbf{v}}{||\mathbf{v}||} \tag{1}$$

3.1.2 Modele

Do klasyfikacji artykułów zostały użyte trzy różne modele, trenowane na dwóch różnych korpusach:

- Routers – zbiór artykułów Reutersa, dostępny w bibliotece NLTK. W skład korpusu, oprócz artykułów wchodzą również krótkie depesze, składające się głównie ze skrótów nazw spółek i kwot transakcji przez nie zawieranych,
- Rzeczywisty – zbiór ok. 400 artykułów uzyskanych przez system w trakcie jego działania.

Wytrenowane i ocenione zostały trzy modele, ich wyniki znajdują się w sekcji 4.4.1:

- reuters – model trenowany na korpusie Reuters,
- reuters+rzeczywisty – model trenowany na obu korpusach,
- rzeczywisty – model trenowany na korpusie rzeczywistych wiadomości.

Preprocessing Przed treningiem korpusy poddane zostały preprocessingowi: usunięte i zastąpione spacją zostały wszystkie symbole nie będące literami lub cyframi.

3.2 GenSim

Do wytrenowania modeli doc2vec użyty został framework Gensim i klasa `Doc2Vec` z modułu `gensim.models.doc2vec`. Umożliwia ona trenowanie modeli doc2vec, jak również wyznaczanie wektorów dla niewidzianych wcześniej dokumentów.

Pewnym utrudnieniem w jej użyciu była bardzo ogólnikowa dokumentacja, jak również brak niektórych oczywistych funkcji, np. policzenie podobieństwa wektorów przy użyciu dostępnej w bibliotece metody `infer_vector` możliwe jest tylko dla wektorów reprezentujących dokumenty ze zbioru treningowego. Tak więc ta funkcjonalność musiała zostać zaimplementowana ręcznie.

Oprócz braku dokumentacji, wynikającego z trwającego jeszcze rozwoju frameworku, Gensim nie sprawił większych kłopotów. Zarówno prędkość treningu, jak i wyznaczania reprezentacji wektorowych jest wystarczająca dla potrzeb projektu. Niestety, funkcje dostępne w klasie `Doc2vec` nie są przystosowane do działania na GPU, którego użycie znacząco przyspiesza trening sieci neuronowych.

4 Badania

W poniższym rozdziale przedstawione zostaną wyniki badań przeprowadzonych przez zespół w celu weryfikacji poprawności działania algorytmu Doc2Vec jako algorytmu umożliwiającego określenie podobieństwa pomiędzy artykułami. Opisane zostaną również zbiory danych uzyskane przez zespół na potrzeby testów.

4.1 Znajdowanie artykułów podobnych jako zagadnienie klasyfikacji

Uzyskanie wiarygodnej, liczbowej miary skuteczności znajdowania podobnych artykułów przez algorytm doc2vec wymagało formalnego zdefiniowania zagadnienia podlegającego ocenie. Zadanie to można zdefiniować jako zadanie klasyfikacji binarnej artykułów na dwie klasy: podobne i niepodobne.

Wymagało to przyjęcia pewnej definicji podobieństwa. Nastręczyło to zespołowi pewnej trudności. Na potrzeby oceny skuteczności algorytmów zespół przyjął podział zbiorów na dwie kategorie: artykułów identycznych oraz artykułów zbliżonych tematyką.

Artykuły identyczne określano jako artykuły, które traktują na temat dokładnie tego samego, konkretnego wydarzenia lub tej samej wypowiedzi danej osoby, jednak pochodzą z różnych źródeł lub w różnym czasie. Przykładem artykułów identycznych mogą być dwa artykuły o podjęciu decyzji o wspólnym występie reprezentacji obu Korei pod jedną flagą na Igrzyskach Olimpijskich w Pyeongchangu.

Artykuły zbliżone przyjęto jako artykuły dotyczące tej samej tematyki lub osoby, lecz nie mające wspólnej, konkretnej genezy - na przykład podzbiór artykułów o klubie piłkarskim Chelsea Londyn.

Wprowadzenie drugiej kategorii wymusiło na zespole zbudowanie sztucznego zbioru umożliwiającego ocenę możliwości rozpoznawania przez algorytm artykułów zbliżonych.

Zdefiniowanie podobieństwa pomiędzy artykułami umożliwiło etykietowanie par artykułów:

$$c(x1, x2) = \begin{cases} 0 & x1, x2 - \text{niepodobne} \\ 1 & x1, x2 - \text{podobne} \end{cases}$$

Formalnie, dalszej klasyfikacji podlegały więc nie same artykuły jako takie, lecz wygenerowane przez moduł analityczny podobieństwo pomiędzy parą artykułów:

$$p(x1, x2), x1 \neq x2 - \text{Podobieństwo pomiędzy artykułami x1 i x2}$$

Podobieństwo to zostało wygenerowane jako iloczyn skalarny przeskalowanych wektorów dwóch artykułów. Ponieważ:

$$p \in [0; 1]$$

Podobieństwo to może być traktowane jako prawdopodobieństwo. Narzuca to analogię wyjścia modułu analitycznego jako klasyfikatora, zwracającego prawdopodobieństwo przynależności dwóch artykułów do klasy podobny. Zespół wykorzystał to do oceny skuteczności algorytmu porównywania artykułów w analogii do oceny klasyfikatora.

4.2 Opis zbioru danych

Na potrzeby przeprowadzenia badań zespół zbudował dwa zbiory artykułów. Zbiór roboczo nazwany "rzeczywistym" powstał przez faktycznne uruchomienie systemu agentowego i zbieranie artykułów z serwisów informacyjnych przez system bez względu na ich tematykę czy datę publikacji. Następnie były oceniane pod kątem podobieństwa tematyki i ręcznie etykietowane przez zespół. Zbiór sztuczny powstał poprzez odwrócenie takiego toku rozumowania - zamiast szukać w gotowym zbiorze podobnych artykułów, zespół wyszukiwał w internecie artykułów traktujących o identycznych bądź zbliżonych tematach.

4.2.1 Zbiór rzeczywisty

Zbiór rzeczywisty powstał poprzez uruchomienie gotowego systemu agentowego i pozyskanie artykułów z kanałów RSS następujących serwisów informacyjnych i mediów:

1. CNN (29 artykułów)
2. BBC (118 artykułów)
3. The Guardian (81 artykułów)
4. The New York Times (114 artykułów)
5. Reuters (30 artykułów)

6. The Washington Times (40 artykułów)

Łącznie pobrano 412 artykułów. Dotyczyły one trzech zakresów tematycznych (*tagów*):

1. Science (70 artykułów)
2. Technology (169 artykułów)
3. Politics (173 artykuły)

Zespół zdecydował o utworzeniu zbioru klasyfikacyjnego na podstawie artykułów dotyczących tagu Politics. W procesie ręcznego przeglądania zbioru, zidentyfikowano łącznie 45 artykułów identycznych dotyczących 15 różnych tematów. W poniższej tabeli wymieniono tytuły, źródła i tematy artykułów identycznych ze zbioru.

Serwis	Tytuł artykułu	Kategoria
Reuters	Congress to vote Thursday for funding bill to avoid government shutdown	Głosowanie w Kongresie
Washington Times	Donald Trump muscles in on GOP strategy to stop shutdown	
Washington Times	James Lankford: Senate will use 'nuclear option' if his plan to limit debate time isn't passed	
CNN	Republicans scramble to try to avert government shutdown as deadline nears	
CNN	Government to shut down in 48 hours: What to watch	
CNN	The Point: Republicans are going to get blamed for a government shutdown. Bigly.	
Reuters	The Point: Republicans are going to get blamed for a government shutdown. Bigly.	Przesłuchanie Bannona
Washington Times	Steve Bannon to be interviewed by Robert Mueller	
CNN	Axios: Bannon had a 'slip-up' in House Intelligence answer	
CNN	Bannon's Hill appearance reveals White House effort to restrict testimony	
CNN	Bannon will do interview with special counsel, avoiding grand jury for now	
New York Times	North Korea, Bannon, Apple: Your Wednesday Evening Briefing	
New York Times	Stephen Bannon, California, Bitcoin: Your Tuesday Evening Briefing	
Reuters	Exclusive: Trump takes hard line on immigration, rejects 'horrible' bipartisan plan	Ustawa imigracyjna Trumpa
Washington Times	White House official: Trump immigration views have evolved	
CNN	McConnell: Trump hasn't said what he wants in immigration bill	
CNN	McConnell: Trump hasn't said what he wants in immigration bill	
Washington Times	Kirstjen Nielsen: Democrats insult Border Patrol with focus on 's—hole' comment	Komentarz Kristjen Nielsen
CNN	DHS Sec. Nielsen: 'I did not and will not lie under oath'	

Guardian New York Times New York Times	 North and South Korean Teams to March as One at Olympics North Korean Orchestra Plans to Perform in South Korea During Winter Olympics	Wspólny występ obu Korei na IO
Guardian CNN CNN	Donald Trump's first year: in his own words - video Obama welcomed Trump to Washington a year ago. They haven't spoken since. What we learned from 365 days of Trump polls	Pierwszy rok Trumpa
Guardian CNN	Tillerson: US military to maintain open-ended presence in Syria – video Tillerson's Syria wish list means US is in it for the longer haul	Tillerson o Syrii
Washington Times CNN	Donald Trump insists the border wall will be built as promised After Kelly said Trump changed his 'attitude' on wall, Trump tweets it 'never changed'	Donald Trump o murze na granicy
CNN CNN	John Kelly: Immigration 'hardass' Kelly on immigration: Trump 'has changed the way he's looked at a number of things'	Kelly o imigracji
Reuters Guardian CNN	Exclusive: Trump accuses Russia of helping North Korea evade sanctions; says U.S. needs more missile defense Trump accuses Russia of violating sanctions to aid North Korea Trump says 'Russia is not helping' with North Korea	Trump o Rosji i Korei
New York Times New York Times	Sex Abuse Case Shadows Pope Francis' Visit to Peru In Chile, Pope Francis Apologizes for 'Irreparable Damage' Caused by Sexual Abuse	Wypowiedź Franciszka
New York Times New York Times	Suicide Bombers Attack Market in Nigeria, Killing at Least 12 Gunmen Kidnap 2 Americans and 2 Canadians in Nigeria	Zamach w Nigerii
New York Times BBC	Facebook to Take Broader Look at Possible Russian Role in Brexit Vote Facebook to reconsider claims of Russian interference in Brexit vote	Facebook o Brexicie
New York Times BBC	Days After Hawaii's False Missile Alarm, a New One in Japan Warning system?	Fałszywy alarm na Hawajach
New York Times BBC BBC	Carillion Collapse Could Lead to Thousands of Job Losses in U.K. Bonuses for Carillion bosses are blocked Carillion's woes	Carillion

Zbiór rzeczywisty został wyeksportowany z bazy danych i zapisany w postaci pliku csv do dalszej analizy. Dla każdej pary artykułów ze zbioru rzeczywistego wygenerowano podobieństwo pomiędzy

nimi. Ponieważ ocenie algorytmu podlega podobieństwo pomiędzy artykułami, utworzony zbiór miał:

1. 56 par artykułów podobnych
2. 14822 pary artykuł niepodobnych

4.2.2 Zbiór sztuczny

Zbiór sztuczny został stworzony ręcznie przez twórców systemu i zawierał dwa rodzaje artykułów.

- artykuły identyczne - o tej samej tematyce,
- artykuły zbliżone - o zbliżonej tematyce

Utworzono 3 grupy artykułów identycznych, każdy po 10 artykułów.

- sztorm w Holandii oraz Niemczech,
- wyrzut cząsteczek z czarnej dziury,
- podniesienie ceny serwisu Amazon Prime,

Artykułów o zbliżonej tematyce również powstały 3 grupy, również po 10 artykułów.

- wiadomości dotyczące Władimira Putina,
- klub piłkarski Chelsea,
- podatność Meltdown w procesorach Intel

4.3 Miary oceny

Zbiór utworzony na podstawie tagu Polityka zawiera zaledwie 56 pozycji o klasie 1 i niemal 15 000 pozycji o klasie 0. Jest to więc zbiór bardzo niezrównoważony. Właściwą metodą oceny algorytmów klasyfikacji dla takich zbiorów nie jest zatem dokładność klasyfikacji (co w opisywanym przypadku i tak nie miałoby sensu), lecz jej czułość i swoistość.

Czułość (ang. *True Positive Rate*) należy w niniejszym projekcie zdefiniować jako zdolność wskazania klasy mniejszościowej - par artykułów podobnych.

Swoistość (ang. *True Negative Rate*) należy zdefiniować jako zdolność poprawnego wskazania klasy większościowej - par artykułów niepodobnych.

Zaproponowana przez zespół analogia stopnia podobieństwa jako prawdopodobieństwa przynależności do klasy par artykułów podobnych nie daje jednak odpowiedzi binarnej. Należy ją uzyskać poprzez progowanie odpowiedzi modułu, uznając, iż jeśli podobieństwo jest powyżej określonego progu, to artykuły są podobne.

Najlepszą metodą oceny tego rodzaju algorytmu jest wykreślenie zależności pomiędzy czułością a swoistością dla wszystkich możliwych progów jako krzywej ROC (ang. *Reverse Operating Characteristic*). Wobec tego wszystkie wyniki badań przeprowadzonych przez zespół będą poniżej opisane z wykorzystaniem tej krzywej.

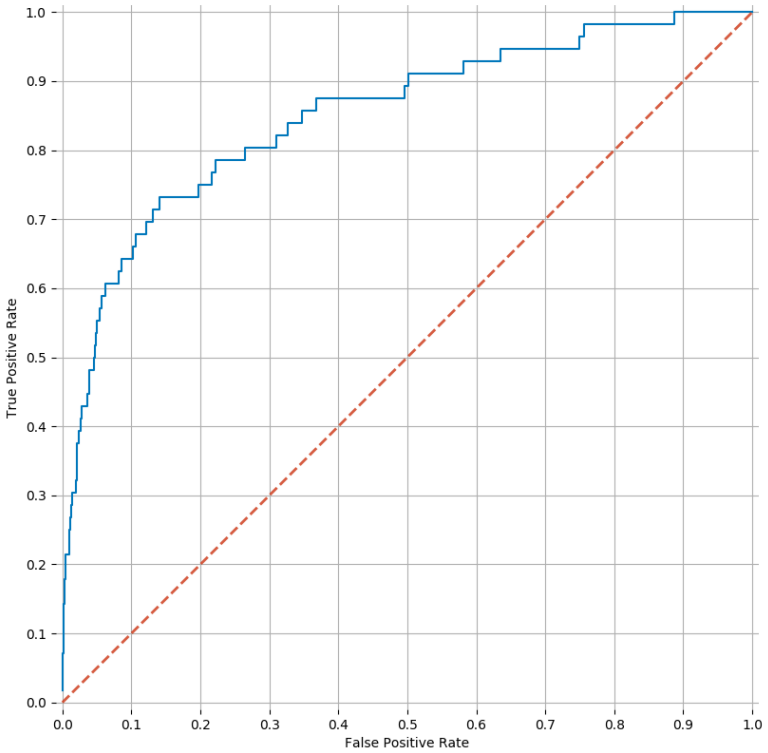
Powszechnie wykorzystywaną miarą liczbową określającą poprawność działania algorytmu, którego wyniki są przedstawione na krzywej ROC, jest pole pod powierzchnią tej krzywej. Im bardziej zbliżone jest ono do wartości 1, tym lepszą klasyfikację algorytm umożliwia.

4.4 Rezultaty

4.4.1 Badania na zbiorze sztucznym

Badaniu podlegał zbiór par artykułów i podobieństwa pomiędzy nimi. Zbiór zawierał:

1. 56 par artykułów podobnych
2. 14822 pary artykuł niepodobnych



Rysunek 3: Krzywa ROC uzyskana na zbiorze rzeczywistym, AUC = 0.85

Uzyskana krzywa ROC świadczy o tym, iż na rzeczywistym zbiorze artykułów algorytm bardzo dobrze radzi sobie z wskazywaniem artykułów o identycznej tematyce. Spośród 56 par artykułów podobnych, zaledwie 4 pary uzyskały stopień podobieństwa niższy niż średni stopień podobieństwa pomiędzy wszystkimi artykułami (0.3). Wszystkie pozostałe artykuły uzyskały znacząco wyższy stopień podobieństwa, co umożliwia dokonanie poprawnej klasyfikacji na podstawie progowania.

Artykuły źle klasyfikowane

Celem znalezienia przyczyn niepowodzeń w klasyfikacji artykułów podobnych, zespół dokonał przeglądu par artykułów, których stopień podobieństwa uzyskany z modułu analitycznego jest niski.

1. Pierwszą analizowaną parą artykułów są artykuły dotyczace wypowiedzi Rexa Tillersona na temat obecności Armii Stanów Zjednoczonych w Syrii:

   (a) CNN - Tillerson's Syria wish list means US is in it for the longer haul

   (b) Guardian - US military to maintain open-ended presence in Syria, Tillerson says

   Zespół upatruje przyczyny niepowodzenia w odmiennej naturze tych dwóch artykułów. Pierwszy spośród nich jest artykułem typowo opiniotwórczym, odnoszącym się do wypowiedzi Tillersona wyłącznie w kontekście Syrii, podczas gdy drugi z nich ma charakter relacji z wypowiedzi, pod koniec zaś odnosi się do posiedzenia Organizacji Narodów Zjednoczonych, czemu poświęcona jest niemal połowa artykułu.

2. Druga niepodobna zdaniem algorytmu para artykułów dotyczy dwóch opiniotwórczych artykułow na temat pierwszego roku prezydentury Donalda Trumpa:

   (a) Donald Trump's first year: in his own words - video

(b) Obama welcomed Trump to Washington a year ago. They haven't spoken since.

Przyczyna niepowodzenia okazała się oczywista po przejrzeniu pierwszego artykułu - jego treść jest w zasadzie wyłącznie nagłówkiem opatrującym nagranie zawierające wypowiedzi Trumpa, zaś same wypowiedzi nie są w nim transkryptowane. Artykuły te dotyczą też tematyki pierwszego roku Trumpa, lecz w odniesieniu do różnych kwestii - drugi koncentruje się na relacji nowego prezydenta z poprzednim. Również kilka kolejnych pozycji spośród listy par artykułów tyczących się identycznej o najniższym stopniu podobieństwa zostało wygenerowanych jako porównania artykułu będącego zapowiedzią filmiku z innymi artykułami o pierwszym roku prezydentury.

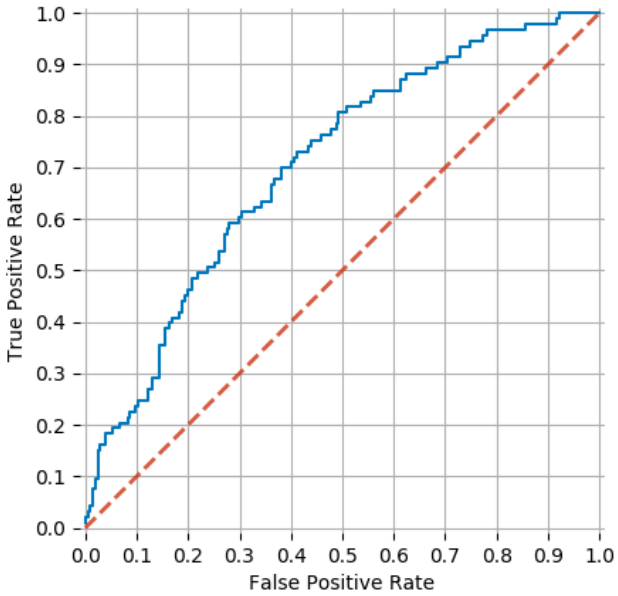
Analiza przypadków słabo sklasyfikowanych ujawniła ułomności uzyskanego zbioru, potwierdzając w gruncie rzeczy zdolności algorytmu do rozpoznawania artykułów podobnych.

Wyniki różnych modeli

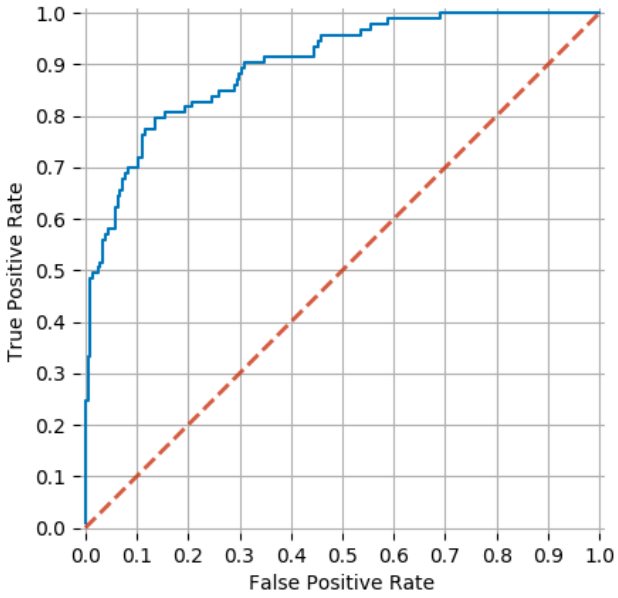
Jako podstawowy model, użyty został model trenowany na korpusie Reutersa, i dla niego przeprowadzone zostały wszystkie powyższe obliczenia. W wyniku pracy systemu udało się zgromadzić zbiór ok. 400 artykułów, który użyty został w celu uzyskania modelu o wyższej dokładności. Eksperymenty te zostały opisane poniżej.

Wyniki modeli opisanych w sekcji 3.1.2 na sztucznym zbiorze testowym, dla artykułów podobnych i zbliżonych:

model	AUC – podobny	AUC – zbliżony
reuters	0.9750	0.7047
reuters+rzeczywisty	0.9893	0.8552
rzeczywisty	1.0000	0.9000



Rysunek 4: Krzywa ROC uzyskana na zbiorze testowym wiadomości zbliżonych dla modelu trenowanego na korpusie Reuters, AUC = 0.7047



Rysunek 5: Krzywa ROC uzyskana na zbiorze testowym wiadomości zbliżonych dla modelu trenowanego na zbiorze rzeczywistym, AUC = 0.9000

Z powyższych wyników można wysnuć dwa wnioski:

Po pierwsze, korpus wiadomości Reutersa jest zbyt różny od rzeczywistego, aby jego użycie w treningu zwiększało wyniki modelu. Zastąpienie go korpusem zbudowanym z wiadomości uzyskanych w trakcie pracy systemu zwiększyło dokładność klasyfikacji. Prawdopodobnie wynika to z różnicy w strukturze wiadomości obu korpusów: korpus Reutersa zawiera, oprócz zwykłych artykułów znaczną liczbę krótkich depeszy prasowych.

Po drugie, wiadomości dotyczące jednego konkretnego wydarzenia (podobne) klasyfikowane są przez system dużo lepiej niż wiadomości zbliżone (dotyczące różnych wydarzeń o podobnej tematyce). Jest to oczekiwany wynik, świadczący o prawidłowym działaniu systemu.

5 Podsumowanie

Zespół w trakcie wykonywania projektu posiadł praktyczną wiedzę zarówno na temat wykorzystania systemów agentowych, jak i niektórych algorytmów przetwarzania języka naturalnego. Realizacja zadań związanych z projektem wymagała przyswojenia sobie przez zespół nie tylko obsługi bibliotek takich jak Gensim czy Akka, ale nawet nowoczesnego języka programowania, jakim jest Scala.

Zespół rozpoznał dzięki temu wady i zalety wykorzystania architektury agentowej systemu oraz konkretnego oprogramowania w postaci Akki. Łatwa budowa modularnego oprogramowania, jego rozszerzalność, prostota z jaką pozwala ona na budowę systemów rozproszonych i bezproblemowa obsługa awarii to cechy, które niewątpliwie przemawiają za dalszym stosowaniem tego rodzaju architektury również poza projektami akademickimi.

Efektem pracy zespołu jest wytworzenie funkcjonalnego systemu umożliwiającego tworzenie bazy danych artykułów z aż pięciu kanałów informacyjnych oraz analizę ich podobieństwa. System ten ma formę rozproszoną, dzięki wykorzystaniu Akki jest mało wrażliwy na awarie węzłów związanych z pobieraniem artykułów. Umożliwia on również poprawne rozpoznawanie artykułów podobnych.

Zespół z wykorzystaniem systemu zbudował bazę danych niemal 500 artykułów na podstawie którego stworzył dwa zbiory badawcze. Posłużyły one do szerokiej weryfikacji poprawności działania systemu. Klasyfikator zbudowany w oparciu o wyniki dostarczane przez system pozwala z bardzo dobrym wynikiem rozpoznawać artykuły poprawne, czego dowodzi miara AUC wynosząca aż 0,88 w przypadku zbioru bazującego na rzeczywistym działaniu systemu. Dla potwierdzenia po-

prawnej jakości rozwiązań wykorzystanych w projekcie, zespół dostarczył w niniejszej dokumentacji rozbudowane analizy i badania przeprowadzone na nim.

Literatura

[1] https://akka.io/.

[2] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053, 2014.