Projekt SAG: analiza sentymentu w mediach społecznościowych

Maciej Lotz, Volodymyr Ostruk, Łukasz Wacławski

25 kwietnia 2017

1 Analiza sentymentu

1.1 Źródło danych i model emocji

Jako źródło danych treningowych posłuży Twitter, z którego dane będą pobierane przez Streaming API. Emocje w ściągniętych tweetach zostaną oznaczone na podstawie zawartych w nich znakach *emoji*. Przyjęty został model 6 podstawowych emocji podany przez Paula Ekmana: złość, obrzydzenie, strach, szczęście, smutek, zaskoczenie.

Po wytrenowaniu modelu klasyfikacji użytkownik będzie mógł podać interesujący go hasztag oraz zakres dat (ograniczony do ostatnich 7 dni ze względu na możliwości API Twittera). W odpowiedzi program przedstawi użytkownikowi statystyki i wykresy nt. emocji wyrażanych przez autorów tweetów w danym hasztagu w danym okresie.

1.2 Algorytm klasyfikacji sentymentu

Jako algorytm klasyfikacji zostanie wykorzystany naiwny klasyfikator bayesowski. Tekst wejściowy będzie dzielony na ngramy. Zaimplementowane zostaną dwie metody wygładzania prawdopodobieństw w modelu: wygładzanie Laplace'a (Lidstone'a) oraz wygładzanie Gooda-Turinga. Zakładmy, że uczenie modelu będzie online.

2 Technologie i typy aktorów

Projekt będzie wykonany w języku Scala z użyciem frameworka Akka. GUI aplikacji zostanie zrobione w ScalaFX, a do rysowania wykresów użyty zostanie pakiet Breeze

Przewidziane są następujące typy aktorów:

- *CategoryModelActor* – będzie odpowiedzialny za przechowywanie danych nt. liczby wystąpień ngramów/dokumentów w danej kategorii, tj. emocji (dla każdej kategorii będzie działał oddzielny aktor). Inni aktorzy będą mu wysyłać kolejne tweety do analizy. Będzie również zwracał na żądanie innych aktorów wygładzone prawdopodobieństwa wystąpienia danego ngramu w przypisanej mu kategorii.

- *NaiveBayesModelActor* – będzie odpowiedzialny za przekazywanie tweetów do analizy odpowiednim *CategoryModelActor* oraz za klasyfikację dokumentów na podstawie informacji udostępnianym mu przez wszystkich *CategoryModelActor*. W celu skalowania aplikacji będzie mogło działać wiele instancji tego typu aktorów jednocześnie.

- *CategoriesRepositoryActor* – będzie odpowiedzialny za tworzenie nowych aktorów typu *CategoryModelActor* jeżeli któryś z aktorów typu. *NaiveBayesModelActor* dostanie do analizy dokument nie widzianej do tej pory kategorii. Będzie również odpowiedzialny za rozesłanie informacji o nowej kategorii.

- *TestingActor* – jego zadaniem będzie ciągłą ewaluacja modelu na podstawie danych testowych podanych przez użytkownika.

- *OnlineTweetStreamer* – będzie na bieżąco ściągał nowe tweety za pomocą Streaming API twittera i podawał je do analizy aktorowi *NaiveBayesModelActor* w celu poprawienia jakości modelu (użytkownik będzie mógł wyłączyć ten typ uczenia). Użytkownik będzie mógł odpalić tyle instancji tego typu aktora ile dostarczy kluczy do API.

- *FileDataStreamer* – możliwa jest sytuacja, w której użytkownik będzie posiadać oetykietowane dane wysokiej jakości (lepsze niż tweety), których dodanie do modelu może polepszyć jakość klasyfikacji. Wtedy *FileDataStreamer* będzie dane wczytywał dane z pliku wskazanego przez użytkownika oraz wysyłał je do *NaiveBayesModelActor*. Uczenie modelu będzie odbywać się w takiej sytuacji z obu źródeł jednocześnie. Użytkownik może również stwierdzić, że interesuje go model wytrenowany tylko na podstawie dostarczonych przez niego danych. Dlatego będzie również istnieć opcja wyłączenia agentów typu *OnlineTweetStreamer*.

- *TweetDownloader* – jego zadaniem będzie ściągnięcie tweetów wg. zadanych przez użytkownika parametrów (hashtag, zakres dat), wysyłanie ich do

- *NaiveBayesModelActor* w celu ich sklasyfikowania oraz agregację wyników. Zebrane w ten sposób statystyki będą następnie wyświetlane użytkownikowi w GUI.