UNIWERSYTET IM. ADAMA MICKIEWICZA W POZNANIU
WYDZIAŁ NAUK SPOŁECZNYCH
KOGNITYWISTYKA

Gracjan Popiółkowski

Algorytmy rozpoznawania sentymentu opinii finansowych

Algorithms for recognizing the sentiment of financial opinions

Poznań 2024

Spis treści

Wprowadzenie 3

1. Prezentacja danych 4
1.1. Zbiór danych 4
1.2. Przygotowanie danych 4

2. Algorytmy analizy sentymentu 5

3. Analiza wyników 6
3.1. Najprostszy model sieci 6
3.2. Porównanie 5 modeli z zmienionym parametrem 7
3.2.1. Model 2 - zwiększony rozmiar embeddingu 7
3.2.2. Model 3 - dodanie warstw dropout 7
3.2.3. Model 4 - zastosowanie LSTM zamiast GRU 7
3.2.4. Model 5 - zwiększenie głębokości modelu 7
3.2.5. Model 6 - zastosowanie konwolucyjnej sieci neuronowej 8
3.3. Porównanie wszystkich 6 modeli 8

Podsumowanie 10

Wprowadzenie

W niniejszym raporcie przedstawiam porównanie wyników analizy sentyment opinii finansowych. Użyję do tego modelu sztucznej inteligencji i porównam z jakimi parametrami działa najlepiej, przy klasyfikacji czy dane zdanie ma wydźwięk pozytywny, neutralny czy negatywny.

ROZDZIAŁ 1

Prezentacja danych

1.1. Zbiór danych

Posługiwałem się zbiorem danych "Financial Sentiment Analysis" udostępnionym przez użytkownika sbhatti na platformie Kaggle. Zbiór obejmuje ponad 5 opinii finansowych, sklasyfikowanych według 3 wydźwięku - pozytywny, neutralny lub negatywny.

1.2. Przygotowanie danych

Tekst poddaje serii operacji przetwarzania wstępnego, by nauka modelu była skuteczniejsza. Oto one:

- Tokenizacja: rozbicie zdań na słowa.
- Konwersja na małe litery: ujednolicenie tekstu poprzez konwersję wszystkich znaków na małe litery.
- Usunięcie interpunkcji: eliminacja znaków interpunkcyjnych z tekstu.
- Usunięcie słów nieznaczących (stop words): wykluczenie słów, które nie niosą istotnego znaczenia semantycznego dla analizy.
- Lematyzacja: sprowadzenie słów do ich podstawowej formy.
- Stemizacja: redukcja słów do ich rdzenia za pomocą algorytmu stemizacji.

ROZDZIAŁ 2

Algorytmy analizy sentymentu

Algorytmy analizy sentymentu tekstu, to metody sztucznej inteligencji i przetwarzania języka naturalnego (NLP), które umożliwiają maszynom zrozumienie, interpretację i klasyfikację subiektywnych informacji zawartych w tekstowych danych. Celem tych algorytmów jest automatyczna ocena tekstu napisanego przez ludzi, aby określić ton emocjonalny, postawę lub uczucia autora tekstu wobec przedmiotu dyskusji. Analiza sentymentu znajduje zastosowanie w różnorodnych dziedzinach, takich jak marketing, obsługa klienta, monitorowanie mediów społecznościowych, finanse i wiele innych, oferując cenne wglądy w opinie, nastroje i preferencje ludzi.

W dalszej części tekstu przedstawię porównanie wyników klasyfikacji do sentymentu po zmianie różnych parametrów.

ROZDZIAŁ 3

Analiza wyników

3.1. Najprostszy model sieci

`Model 1` - to najprostszy model sieci neuronowej, zdefiniowany za pomocą interfejsu programowania aplikacji (API) Keras w TensorFlow, jest przykładowy model do analizy sentymentu tekstu z wykorzystaniem trzech głównych warstw: Embedding, GRU i Dense.

- `Warstwa Embedding` - Ta warstwa przekształca indeksy słów na gęste wektory o stałej długości (u nas 10). Maksymalną liczbę unikalnych słów, które model może przetwarzać, ustawiłem na 50000. Wymiarowość przestrzeni wektorowej, do której słowa będą osadzane, jest równa 10.
- `GRU(16)` - GRU to rodzaj warstwy rekurencyjnej, która ułatwia modelowi zapamiętywanie długoterminowych zależności w danych sekwencyjnych dzięki mechanizmom bramek. Liczba 16 oznacza liczbę jednostek (neuronów) w warstwie GRU.
- `Warstwa Dense` - jest to klasyczna warstwa gęsto połączona (pełna), która na wyjściu generuje rozkład prawdopodobieństwa dla trzech klas (stąd 3). Używa funkcji aktywacji softmax, aby przekształcić wartości wyjściowe w prawdopodobieństwa, dzięki czemu model może dokonać klasyfikacji do jednej z trzech kategorii sentymentu (pozytywny, negatywny, neutralny).
- `Kompilacja` - model jest kompilowany z optymalizatorem adam, funkcją straty sparse_categorical_crossentropy (odpowiednia dla klasyfikacji wieloklasowej, gdzie etykiety są liczbami całkowitymi), oraz metryką accuracy do śledzenia dokładności klasyfikacji.

Modele będę porównywał według wielkości `accuracy` i `val_accuracy`, czyli zmiennej oceniające skuteczność klasyfikacji na zbiorze treningowym i walidacyjnym (testowych). `Val_accuracy` mierzy, jak dobrze model generalizuje się na nowych, wcześniej niewidzianych danych. To ważne, aby sprawdzić, czy model nie jest nadmiernie dopasowany (overfitting), co może zdarzyć się, gdy dokładność treningowa jest wysoka, ale dokładność walidacyjna jest niższa. Wyniki w modelu startowym:

- `Accuracy` = 0.9048
- `Val_accuracy` = 0.6544

Czyli skuteczność klasyfikacji na nowych danych wynosi 65,44%. Co jest bardzo dobrym wynikiem przy 3 grupach klasyfikacyjnych.

3.2. Porównanie 5 modeli z zmienionym parametrem

W drugiej części analizy przedstawię wyniki 5 modeli, w których zmieniłem po jednym kluczowym parametrze z modelu startowego.

3.2.1. Model 2 - zwiększony rozmiar embeddingu

W modelu 2 zdecydowałem się zwiększenie rozmiaru embeddingu z 10 do 50, co oznacza, że każde słowo będzie reprezentowane przez wektor o większej liczbie wymiarów. Teoretycznie pozwala to na bogatsze reprezentacje semantyczne słów, uchwycenie bardziej złożonych relacji między nimi oraz lepsze oddzielenie słów o różnych znaczeniach. Skuteczność tego modelu wyniosła:

- `Accuracy` = 0.9180
- `Val_accuracy` = 0.6270

Model ten odrobinę lepiej poradził sobie na danych treningowych ale już gorzej na testowych.

3.2.2. Model 3 - dodanie warstw dropout

Dalej wprowadziłem funkcję dropout (o wartości 0.5) do modelu podstawowego, by sprawdzić czy tutaj mogę polepszyć wyniki redukując przeuczenie. Oto wyniki:

- `Accuracy` = 0.9155
- `Val_accuracy` = 0.6416

3.2.3. Model 4 - zastosowanie LSTM zamiast GRU

Zmiana z GRU na LSTM może poprawić zdolność modelu do modelowania zależności długoterminowych w danych, ale także zwiększyć złożoność obliczeniową i ryzyko przeuczenia. Oto wyniki:

- `Accuracy` = 0.9150
- `Val_accuracy` = 0.6493

3.2.4. Model 5 - zwiększenie głębokości modelu

Dodanie dodatkowych warstw do modelu (czyli zwiększenie jego głębokości) może pozwolić na uchwycenie bardziej złożonych i abstrakcyjnych reprezentacji danych. Teoretycznie, głębsze modele mają większą zdolność do nauki skomplikowanych wzorców, co może przekładać się na lepszą wydajność. Oto wyniki:

- `Accuracy` = 0.9163
- `Val_accuracy` = 0.6390

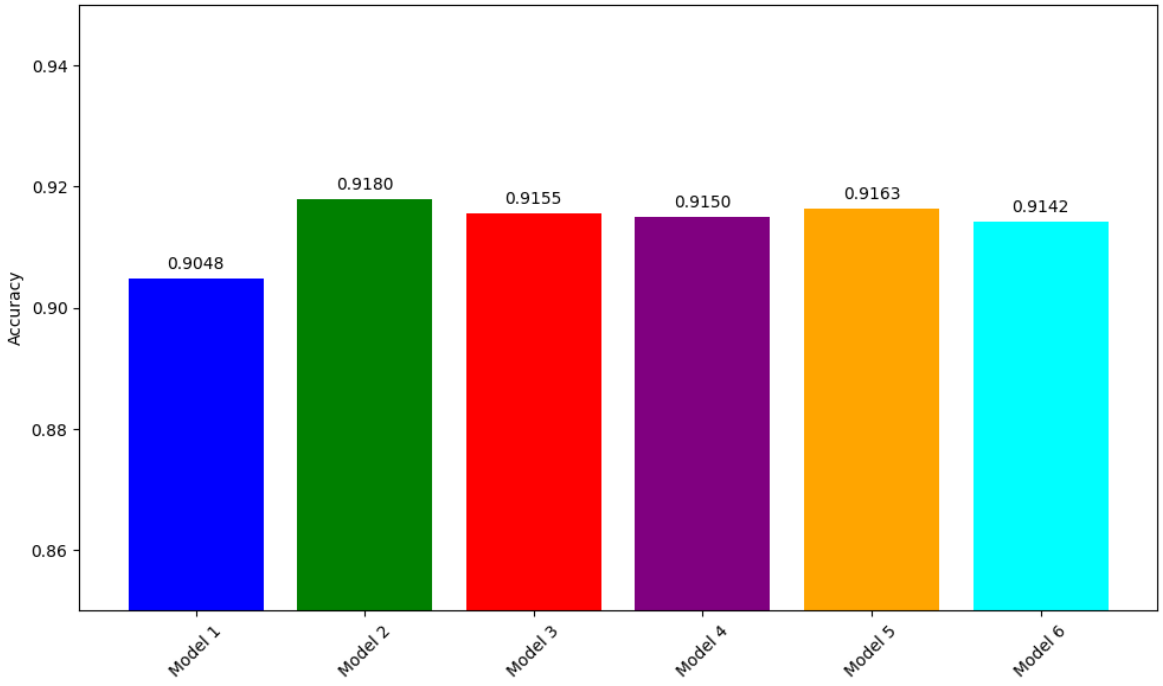
3.2.5. Model 6 - zastosowanie konwolucyjnej sieci neuronowej

Ostatecznie dodałem warstwę konwolucyjną. CNN są tradycyjnie stosowane w przetwarzaniu obrazów, ale okazały się również skuteczne w zadaniach NLP dzięki swojej zdolności do wykrywania lokalnych wzorców w danych.

- `Accuracy` = 0.9142
- `Val_accuracy` = 0.6356

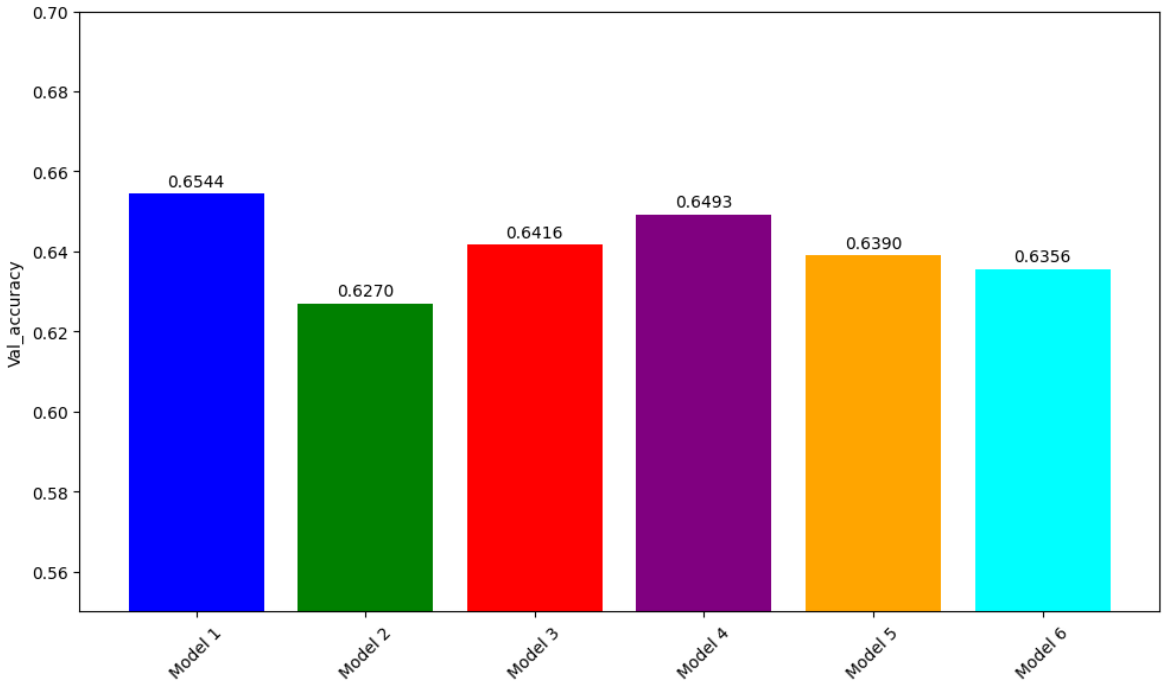
3.3. Porównanie wszystkich 6 modeli

W trzeciej części mojej analizy spojrzymy sobie jak wypadają wyniki wszystkich sześciu modeli na jednym wykresie, jeżeli chodzi o accuracy:



Rysunek 3.1: Porównanie wszystkich 6 modeli pod względem accuracy.

i jeżeli chodzi o val_accuracy:



Rysunek 3.2: Porównanie wszystkich 6 modeli pod względem val_accuracy.

Jak widać jeżeli chodzi o dokładność na zbiorze testowym to nie udało mi się zupełnie znaleźć lepszego rozwiązania niż podstawowy model 1. Wszystkie modele od 2 do 6 były lepsze na zbiorach treningowych o około 1 punkt procentowy.

Podsumowanie

Reasumując, w mojej pracy użyłem różnych modeli sieci do klasyfikacji sentymentu, najlepszym modelem pod względem accuracy okazał się model podstawowy:

- `Accuracy` = 0.9048
- `Val_accuracy` = 0.6544

Najlepszy wynik w val_accuracy uzyskał model 2 z zwiększonym embeddingiem:

- `Accuracy` = 0.9180
- `Val_accuracy` = 0.6270

Skonstruowane modele uzyskały wyniki około 65% skuteczności klasyfikacji sentymentu w zdaniach ustawionych na grupę testową. Przy trzech możliwych klasach: pozytywny, neutralny i negatywny, szansa na losowe przyporządkowanie sentymentu wynosi 33,(3)%. Mój model ma więc prawie dwa razy większą szansę na poprawne przyporządkowanie. Jednak z pewnością przydałyby się kolejne testy, by sprawdzić, czy taki algorytm sztucznej inteligencji jest w stanie osiągnąć więcej. Na pewno też większa baza danych mogłaby wpłynąć na skuteczność modelu.