Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH

przedmiot
Wprowadzenie do przetwarzania języka naturalnego

Analiza wydźwięku w języku angielskim na bazie recenzji produktów

Dokumentacja wstępna

Olga Sapiechowska, Maciej Marcinkiewicz
Numer albumu 302687, 300171

prowadzący
dr inż. Piotr Andruszkiewicz

WARSZAWA 14 grudnia 2022

Spis treści

1. Wstęp

1.1. Temat projektu

Projekt polega na pobraniu opinii w języku angielskim o produktach z kategorii: proszki do prania kolorów, tabletki do zmywarki i kubki termiczne oraz zrealizowania modelu wykrywającego wydźwięk: pozytywny, neutralny, negatywny. Składa się z następujących zadań do zrealizowania:

1. Pobranie opinii z portali internetowych i przygotowanie korpusu.
2. Stworzenie modelu.
3. Stworzenie drugiego modelu wykorzystującego dane dostępne w literaturze w ramach dotrenowania wykorzystywanego pretrenowanego modelu, np. BERT.

1.2. Definicja problemu

Wykrywanie wydźwięku/analiza sentymentów pozwala na zautomatyzowane określanie, czy dany tekst wyraża pozytywne, negatywne, czy neutralne zdanie na temat zadanego produktu lub konceptu (w najprostszym wariancie - istnieją również wersje realizujące np. wykrywanie emocji czy wyodrębnianie konkretnych aspektów produktu, które w szczególności interesują klientów, natomiast są one poza zakresem realizowanego projektu). Zastosowanie analizy sentymentów w kontekście biznesowym znacznie przyspiesza wyciąganie wniosków z nieoznaczonych zbiorów danych takich jak recenzje oferowanych produktów, wyniki ankiet satysfakcji, zgłoszenia do pomocy technicznej, komentarze na mediach społecznościowych, itp. - umożliwia dostrzeżenie pewnych wzorców w zbiorach o dużej objętości bez konieczności przeglądania wszystkich tekstów i oznaczania ich „ręcznie".

Celem projektu jest zbudowanie dwóch modeli wyznaczających wydźwięk zadanego tekstu, przeprowadzenie testów rozwiązania, oceny jakości modeli oraz ich porównanie, a następnie opisanie spostrzeżeń i wniosków. Danymi, na których powinny się uczyć i operować modele, mają być własnoręcznie pozyskane zbiory recenzji produktów z trzech kategorii z dowolnego portalu służącego do sprzedaży internetowej. Architektura rozwiązania powinna być oparta na opisanych w literaturze metodach analizy sentymentów. Modele powinny operować na recenzjach w języku angielskim. Program będący rezultatem ma w zamyśle być „gotowy do użytku", np. przez firmę zajmującą się dystrybucją proszków do prania kolorów i zamawiającą analizę opinii klientów na temat nowo wprowadzonego na rynek produktu.

Niniejszy dokument omawia znalezione w literaturze metody i algorytmy rozwiązania problemu, opisuje proponowaną architekturę systemu, a następnie prezentuje sposób pozyskania i obróbki danych oraz szkielet planowanej implementacji projektu.

2. Metody analizy sentymentów

2.1. Rodzaje zadań

Podstawowym i najpowszechniejszym zadaniem w ramach analizy wydźwięku jest wykrywanie emocji, jakie niesie ze sobą dany dokument lub zdanie – pozytywnych lub negatywnych, rzadziej neutralnych. Ten rodzaj zadań jest tematem niniejszego projektu. Innymi powszechnymi zadaniami jest klasyfikacja tekstów jako subiektywne lub obiektywne oraz ocena dokumentów jako istotne bądź nieistotne względem danego tematu.

Powyższe zagadnienia są problemami dychotomicznymi, nie licząc wariantu wykrywania emocji z uwzględnieniem neutralnych wypowiedzi, ale wieloklasowe problemy również mogą należeć do kręgu wykrywania wydźwięku. Modele wieloklasowe mogą oceniać między innymi to, jakiej kwestii dotyczy dany dokument.

2.2. Wykorzystywane techniki

2.2.1. Metody bazujące na wiedzy

Do wykrywania wydźwięku można podejść metodami bazującymi na wiedzy oraz metodami statystycznymi (uczenie maszynowe). Te pierwsze wykorzystują wcześniej zdobytą wiedzę na temat znaczenia i wydźwięku danych słów. Obecność jednoznacznie nacechowanych słów przeważa za nadaniem dokumentowi odpowiedniej kategorii, na przykład obecność słów „zły” i „zdenerwowany” wskazuje na potencjalny negatywny wydźwięk tekstu.

2.2.2. Uczenie maszynowe

Metody uczenia maszynowego z kolei analizują teksty, którym zostały wcześniej przypisane etykiety z danymi emocjami, a następnie na bazie wyuczonych wzorców dokonują predykcji wskazując klasę podanych do modelu danych. Są do tego wykorzystywane zarówno klasyczne metody uczenia z nadzorem takie jak maszyny wektorów nośnych, losowy las decyzyjny i naiwny klasyfikator Bayesa, ale także sieci neuronowe, między innymi sieci rekurencyjne, perceptrony wielowarstwowe, czy nawet używane głównie do przetwarzania obrazów splotowe sieci neuronowe **kim2014**.

Uczenie maszynowe wymaga, aby dane były reprezentowane w sposób numeryczny. Dane można reprezentować na przykład za pomocą bag-of-words - multizbioru słów, zawierającego jedynie statystykę wystąpień słów, ale także za pomocą word embeddings, czyli osadzeń słów w wektory. Metody word embedding pozwalają na zapisanie znaczeń słów – dwa słowa są tym bliższe sobie, im bliżej są ich wektory w przestrzeni liniowej. Powszechnie stosuje się wcześniej wytrenowane modele word embeddingu takie jak word2vec[1], GloVe[2] czy fastText[3].

Jeszcze bardziej zaawansowane metody próbują analizować kontekst czy podmiot w danej wypowiedzi na podstawie analiz gramatycznych.

Z kolei najnowszym osiągnięciem w dziedzinie są architektury transformatowe[4]. Są skonstruowane w sposób podobny do sieci rekurencyjnych, ale w przeciwieństwie do nich przetwarzają wszystkie dane wejściowe na raz oraz wykorzystują mechanizm atencji. Do takich modeli należą BERT[5] czy GPT[6]. Te modele mogą służyć w wielu problemach związanych z przetwarzaniem języka naturalnego, włącznie z analizą wydźwięku. Mogą być również zastosowane jako warstwa emebddingu do innych architektur, zamiast dedykowanych temu modeli jak wspomniane wcześniej embeddery.

2.2.3. Metody hybrydowe

Modele uczenia maszynowego mogą być wspierane przez metody oparte na wiedzy. Wiedza może pochodzić na przykład z ontologii oraz z sieci semantycznych.

2.3. Przykłady istniejących architektur

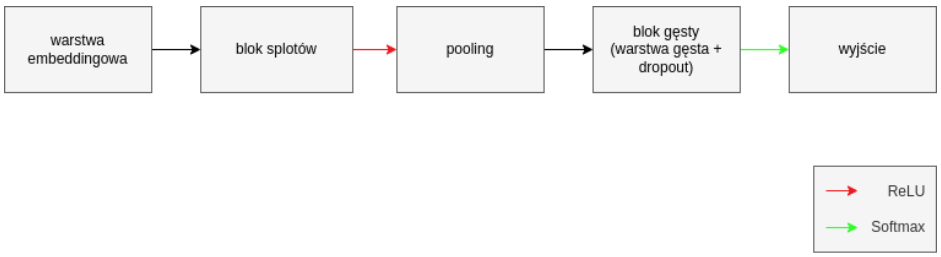
Jedną z najbardziej znanych architektur sieci neuronowych do klasyfikacji tekstu jest fastText[7]. Została stworzona przez zespół badawczy Facebooka, następnie projekt wyewoluował w bibliotekę zawierającą także narzędzia do word embeddingów. Jest to płytka sieć neuronowa. Składa się jedynie z warstwy embeddingowej, warstwy gęstej (ukrytej) oraz warstwy wyjściowej zakończonej funkcją softmax. Zdecydowaną zaletą tej architektury jest jej szybkość. Jest w stanie osiągnąć dokładność architektur opartych na sieciach rekurencyjnych przy znacznie krótszym czasie treningu.

Nieco mniej konwencjonalną metodą analizy wydźwięku jest stosowanie splotowych sieci neuronowych (CNN). Yoon Kim w swojej pracy „Convolutional Neural Networks for Sentence Classification”[8] porównał proste modele oparte na sieci złożonej z warstwy embeddingowej, z wektorami uzyskanymi z word2vec, jednej warstwy splotu oraz wyjściowej warstwy gęstej zakończonej softmaxem. Modele stworzone przez autora artykułu w większości przypadków miały przewagę nad innymi modelami. Ponadto została przez niego podkreślona istotność wykorzystania wyuczonych wcześniej word embeddingów – wariant CNN pozbawiony word2vec radził sobie najgorzej.

3. Opis proponowanego rozwiązania

Proponujemy rozwiązanie oparte na artykule „Convolutional Neural Networks for Sentence Classification”[8]. Sieć zaczyna się od warstwy embeddingowej dostarczającej wektory słów. Po niej następuje przetwarzanie przez blok konwolucyjny składający się z jednowymiarowych warstw splotowych, których wielkość kernela będzie zależała od tego, jakie n-gramy będzie analizować , tzn. wielkość kernela odpowiada wielkości n-gramu. Warstwy konwolucyjne będą zakończone funkcją aktywacji ReLU. Blok może składać się z wielu warstw – określenie ilości analizowanych n-gramów będzie pozostawione użytkownikowi sieci. Po splocie następuje warstwa poolingowa realizująca max-over-time pooling. Ostatni blok sieci stanowi warstwa gęsta z warstwą dropout (w celach regularyzacji), zakończony jest softmaxem.

Jako optymalizator zostanie zastosowany Adam, zaś funkcją kosztu z racji wieloklasowości problemu będzie entropia skrośna.



Rysunek 3.1. Proponowana architektura klasyfikatora opartego warstwach splotowych.

Drugi model do celów porównawczych będzie oparty o wektor słów z pretrenowanego modelu transformatorowego BERT[5]. Zmiana tyczy się tylko warstwy embeddingowej – będą do niej wprowadzone inne wektory.

4. Projekt implementacji

4.1. Skrypt do pozyskiwania danych z serwisu Amazon

Przed przystąpieniem do trenowania modeli analizy wydźwięku należy zebrać odpowiednio liczny zbiór danych, które posłużą za dane testowe i treningowe dla implementowanego algorytmu.

Ze względu na fakt, że istnieje wiele narzędzi pomagających w pobieraniu wpisów zintegrowanych z tym portalem, a także na łatwość przekładania liczby gwiazdek przyznanych produktowi przez użytkownika na wydźwięk pozytywny, negatywny bądź neutralny, zdecydowano się trenować model na danych pochodzących z amerykańskiej wersji sklepu internetowego Amazon.

Wybrano po kilka-kilkanaście produktów ze zadanych trzech kategorii i wyszukano je w serwisie Amazon. Następnie ich strony główne, zawierające recenzje stanowiące docelowy zestaw danych, pobrano w formie pliku HTML. W tym celu wykorzystano interfejs **ScraperAPI**[9], obsługujący serwery proxy, przeglądarki oraz CAPTCHA i tym samym pozwalający na uzyskanie HTML z dowolnej strony internetowej o znanym adresie URL za pomocą prostego wywołania w dowolnym języku. Zastosowanie takiego interfejsu lub innego, równoważnego narzędzia jest kluczowe w przypadku pobierania danych z serwisu, który implementuje zabezpieczenia przeciwko botom - inaczej żądania są blokowane.

Narzędzie, które zostało wykorzystane w celu wydobycia z otrzymanych w ten sposób dokumentów istotnych informacji to **Beautiful Soup**[10] - biblioteka dostępna dla języka Python. Pozwala ona na przeszukiwanie plików HTML i XML przy pomocy prostych zapytań. Przykładowo, poniższy fragment kodu znajduje wszystkie elementy HTML będące znacznikiem span i dla których atrybut data-hook ma ustawioną wartość review-body:

```
soup.find_all("span", {"data-hook": "review-body"})
```

Każdy z otrzymanych w poprzednim kroku plików przeszukano i dla każdego produktu wydobyto treści wszystkich widocznych na stronie recenzji razem z przydzielonymi gwiazdkami. Następnie przystąpiono do oznaczania sentymentów. Przyjęliśmy założenie, że ocena wynosząca 4 lub 5 gwiazdek oznacza wydźwięk pozytywny, ocena 3 – wydźwięk neutralny, a przyznanie 1 albo 2 gwiazdek wskazuje na wydźwięk negatywny. Tak oznaczone recenzje (treść – sentyment) zostały zgrupowane według kategorii produktu i zapisane w pliku w formacie CSV.

4.1.1. Preprocessing

Preprocessing zostanie wykonany przy użyciu biblioteki **spaCy**[11]. Jednak zamiast wykorzystywać ją bezpośrednio, zostanie ona użyta wewnątrz modułu tokenizatora biblioteki **PyTorch**[12]. PyTorch będzie podstawowym narzędziem do utworzenia modelu sieci neuronowej, więc taka integracja narzędzi znacznie ułatwi konstrukcję systemu.

Dane będą załadowane do klasy dziedziczącej po klasie Dataset z PyTorcha. Klasa ta stanowi API dla klasy Dataloader, która ułatwia wprowadzanie treningowych, walidacyjnych oraz testowych danych do sieci.

4.2. Modele wykrywające wydźwięk

Jak zostało wspomniane wyżej, model zostanie zbudowany przy pomocy biblioteki PyTorch[12]. Biblioteka ta pozwala na tworzenie architektur sieci neuronowych od zera przy pomocy gotowych funkcji implementujących dane rodzaje warstw sieci. Poza narzędziami do tworzenia modeli, PyTorch dostarcza funkcje służące do optymalizacji, wyliczania wartości funkcji kosztu, czy regualaryzacji.

Aby stworzyć model z modelem BERT służącym za źródło word embeddingu również pomocny będzie PyTorch, a dokładniej jego komponent **torchtext**. Zawiera wszelkie narzędzia służące do przygotowania danych będących tekstem w języku naturalnym, w tym także pretrenowane modele word embeddingu, klasyfikatory oraz modele transofrmatorowe.

Do ewaluacji zostanie wykorzystana biblioteka **TorchMetrics**[13]. Posiada zdefiniowane funkcje ze wszystkimi powszechnie stosowanymi metrykami. Jej użycie z modelami PyTorch jest niezwykle proste, ponieważ wykonuje operacje na tensorach będących podstawową jednostką danych w PyTorch.

Bibliografia

[1] T. Mikolov, K. Chen, G. Corrado i J. Dean, *Efficient Estimation of Word Representations in Vector Space*, 2013. DOI: 10.48550/ARXIV.1301.3781. adr.: https://arxiv.org/abs/1301.3781.

[2] J. Pennington, R. Socher i C. Manning, "GloVe: Global Vectors for Word Representation", w *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, paź. 2014, s. 1532–1543. DOI: 10.3115/v1/D14-1162. adr.: https://aclanthology.org/D14-1162.

[3] P. Bojanowski, E. Grave, A. Joulin i T. Mikolov, *Enriching Word Vectors with Subword Information*, 2016. DOI: 10.48550/ARXIV.1607.04606. adr.: https://arxiv.org/abs/1607.04606.

[4] A. Vaswani, N. Shazeer, N. Parmar i in., *Attention Is All You Need*, 2017. DOI: 10.48550/ARXIV.1706.03762. adr.: https://arxiv.org/abs/1706.03762.

[5] J. Devlin, M.-W. Chang, K. Lee i K. Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, 2018. DOI: 10.48550/ARXIV.1810.04805. adr.: https://arxiv.org/abs/1810.04805.

[6] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever i in., "Improving language understanding by generative pre-training", 2018.

[7] A. Joulin, E. Grave, P. Bojanowski i T. Mikolov, *Bag of Tricks for Efficient Text Classification*, 2016. DOI: 10.48550/ARXIV.1607.01759. adr.: https://arxiv.org/abs/1607.01759.

[8] Y. Kim, *Convolutional Neural Networks for Sentence Classification*, 2014. DOI: 10.48550/ARXIV.1408.5882. adr.: https://arxiv.org/abs/1408.5882.

[9] S. LLC, *ScraperAPI*. adr.: https://www.scraperapi.com (term. wiz. 14. 12. 2022).

[10] L. Richardson, *Beautiful Soup*. adr.: https://www.crummy.com/software/BeautifulSoup (term. wiz. 14. 12. 2022).

[11] M. Honnibal i I. Montani, "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing", To appear, 2017.

[12] A. Paszke, S. Gross, F. Massa i in., "Pytorch: An imperative style, high-performance deep learning library", *Advances in neural information processing systems*, t. 32, 2019.

[13] L.-A. et al., *TorchMetrics*. adr.: https://torchmetrics.readthedocs.io (term. wiz. 14. 12. 2022).