Detekcja fake news

Radomir Krawczykiewicz, Grzegorz Wątor

8 stycznia 2020

1 Wstęp

Media społecznościowe są obecnie głównym źródłem pozyskiwania wiadomości. Ze skutkiem wypierają tradycyjne media takie jak telewizja czy prasa, a to wszystko dzięki bezpłatności, szybkości dostępu oraz powszechności, którą oferują. Wydają się one być doskonałym narzędziem dla osób indywidualnych do publikowania oraz konsumowania zamieszczonych informacji. Jednak w parze z tymi zaletami idą również poważne wady czy nawet zagrożenia. Jednym z takich zagrożeń jest coraz bardziej rozpowszechniające się zjawisko fake news.

W związku z faktem, że w mediach społecznościowych nie ma organu regulacyjnego, jakość wiadomości rozpowszechnianych jest często niższa niż w tradycyjnych źródłach wiadomości. Innymi słowy, media społecznościowe pomagają w rozgłaszaniu tzw. fake news, czyli fałszywych wiadomości, które umyślnie wprowadzają ludzi w błąd. Fake news mogą tak samo dotyczyć indywidualne osoby jak i społeczeństwo jako całość. W związku z olbrzymimi możliwościami manipulacji, jakie wynikają z propagowania fake news, kluczowym zagadnieniem jest potrzeba wykrywania fałszywych wiadomości.

Dużym problemem obecnych mediów społecznościowych jest system rekomendacji. Im częściej będziemy wchodzić w interakcje z fake newsami, tym częściej będą się nam one pokazywać. Tym samym wpadniemy w błędne koło, które może doprowadzić do efektu tzw. fake news echo chamber. Efekt ten polega na tym, że są nam prezentowane wyłącznie fake newsy, które odpowiadają jednemu punktu widzenia, przez co możemy nie dostrzec innych opinii.

Wykrywanie fake newsów w mediach społecznościowych jest unikalnym wyzwaniem, które doczekało się już paru prób rozwiązania. Jedną z trudności w skonstruowaniu takiej solucji jest fakt, że obecnie trudnym jest znalezienie odpowiedniego zbioru danych.

2 Cel projektu

Celem projektu jest skonstruowanie mechanizmu, który będzie w stanie dokonać klasyfikacji fake news. Jako klasyfikacje rozumiemy dobranie binarnej etykiety, która będzie oznaczać czy dany artykuł jest fake newsem czy nie.

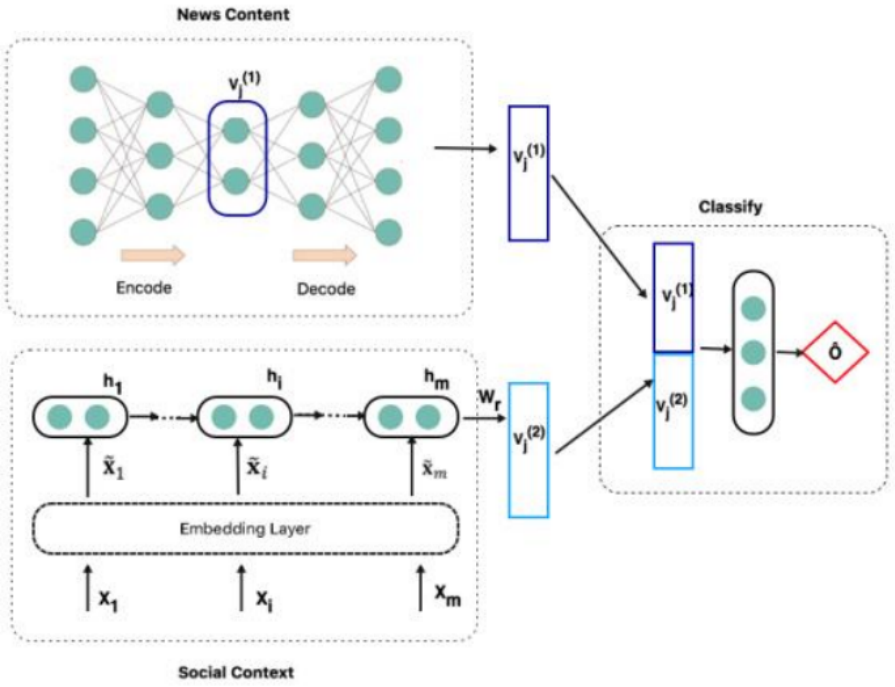
Przedstawione rozwiązanie będzie ściśle porównywane z rozwiązaniem zaproponowanym przez grupę: K.Shu, D. Mahudeswaran, S Wang, D. Lee i H. Liu, która w licznych artykułach prezentują rozwiązania tego problemu. Posługują się oni samodzielnie skonstruowanym zbiorem danych FakeNewsNet, na którym również będziemy ewaluować swoje rozwiązanie.

3 Powiązane prace

W artykule Fake News Detection on Social Media: A Data Mining Perspective [13], napisany przez grupę Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tan, Huan Liu, szeroko opisano problemy fake newsów jak i ich definicje. Zaproponowane zostało również wiele czynników na podstawie, których mogłaby odbyć się klasyfikacja fake newsów. Twórcy odseparowali treść artykułu od jego wpływu na społeczność. Na podstawie zawartości tekstu możemy na przykład zbadać jego poprawność merytoryczną lub styl. W przypadku społeczności możemy zbadać jak szybko w mediach społecznościowych rozszerzała się wieść na temat artykułu oraz jacy użytkownicy wchodzi w nim w interakcje.

K. Shu, D. Mahudeswaran i H. Liu w swojej pracy FakeNewsTracker: A Tool for Fake News Collection, Detection, and Visualization [10] opisują zaprojektowany przez siebie system do wykrywania fałszywych wiadomości. Metoda która opracowali nosi nazwę SAF (Social Article Fusion) i jest oparta na technikach uczenia głębokiego. Jej schemat został przedstawiony na rysunku 1, a jej ogólny zamysł polega na podzieleniu danych na dwie części: artykuły

oraz powiązany kontekst społeczny. Artykuły są reprezentowane jako wektory one-hot encoded, których wymiary są później zmniejszane przez sieć autoasocjacyjną (autoenkoder). Kontekst społeczny w postaci tweetów został przetworzony przez sieć rekurencyjną Long Short-Term Memory (LSTM). Zostały zaproponowane 3 modele, SAF /S(uwzględnia tylko treść artykułu), SAF /A(uwzględnia tylko aspekt społeczny) oraz SAF, który łączy oba podejścia po przez konkatenacje obu wektorów.



Rysunek 1: Schemat aritektury SAF

W artykule [11] K.Shu, D Mahudeswaran, S. Wang, D.Lee i H.Liu szerzej omawiają swój zbiór danych wyniki FakeNewsNet oraz porównują wyniki swoich działań. Swoje eksperymenty przeprowadzili zarówno dla podzbioru PolitiFact jak i GossipCop. Nowymi podejściami w stosunku do opisanych w poprzednim paragrafie są metody skupione w klasyfikowaniu artykułu na podstawie zawartości artykułu. Zawartość przekształcają jako wektory onehot encoded a następnie stosują standardowe techniki uczenia maszynowego

takie jak maszyna wektorów nośnych (SVM), regresja logistyczna (LR), naiwny klasyfikator Bayesowki (NB) oraz sieci konwolucyjne (CNN). Dla maszyny wektorów nośnych, regresji logistycznej oraz naiwnego klasyfikatora Bayesowskiego został użyty domyślny zestaw ustawień dostarczony w pakiecie scikit-learn a parametry nie były dostrajane. Dla konwolucyjnych sieci neuronowych została użyta implementacja opisana w repozytorium *Convolutional Neural Network for Text Classification in Tensorflow* [1]. Domyślne parametry tej implementacji również nie zostały zmienione. Rezultaty z wszystkich wymienionych wyżej metod zostały opracowane standardowymi wskaźnikami takimi jak *accuracy, precision, recall, F1.*

W pracy Hierarchical Propagation Networks for Fake News Detection: Investigation and Exploitation [12] został wykorzystany model wyszukiwania propagandy(HPNF) do klasyfikacji czy dany artykuł jest prawdziwy czy nie. Natomiast w pracy The Role of User Profiles for Fake News Detection [14] zostały przeanalizowane profile użytkowników i na podstawie ich powstał klasyfikator(UPF).

4 Zbiór danych

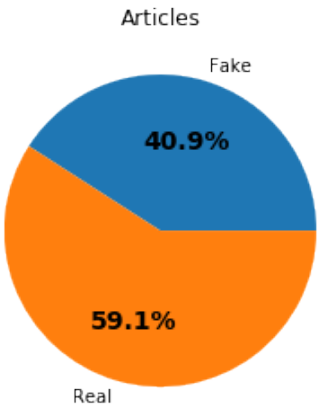
FakeNewsNet zawiera artykuły z serwisu Politifact o charakterze politycznym, oraz artykuły z serwisu Gossipcop o charakterze plotkarskim. Dodatkowo, każda z publikacji zawiera referencje do listy tweetów, które są z nim powiązane.

Niestety, przez politykę prywatności Twitter'a wpisy nie mogą być udostępnione w zbiorze danych, a trzeba je samodzielnie pobrać. Aby to zrobić należy założyć konto developerskie i uruchomić skrypt który przygotowali twórcy[3].

Twórcy umożliwi łatwe konfigurowanie skryptu, tak aby pobierał nie tylko informacje o tweetach, ale również o jego użytkownikach i jego ostatnich tweetach. Niestety ze względu na dynamikę tych zmian, uznaliśmy że używanie tego typu danych, może być nie miarodajne, ponieważ dane które użytkownik udostępnił rok temu mogą różnić się istotnie od obecnych danych.

Niestety treści artykułów nie zostały również zawarte w zbiorze danych, ale skrypt dostarczony od twórców pobiera zarówno treści artykułów jak i powiązane z nim tweety.

W naszej pracy skupiliśmy się na podzbiorze Politifact. Politifact składa się z 1056 artykułów, z czego 432 są uznane za fake news. Podział danych przedstawiliśmy na obrazku 2.



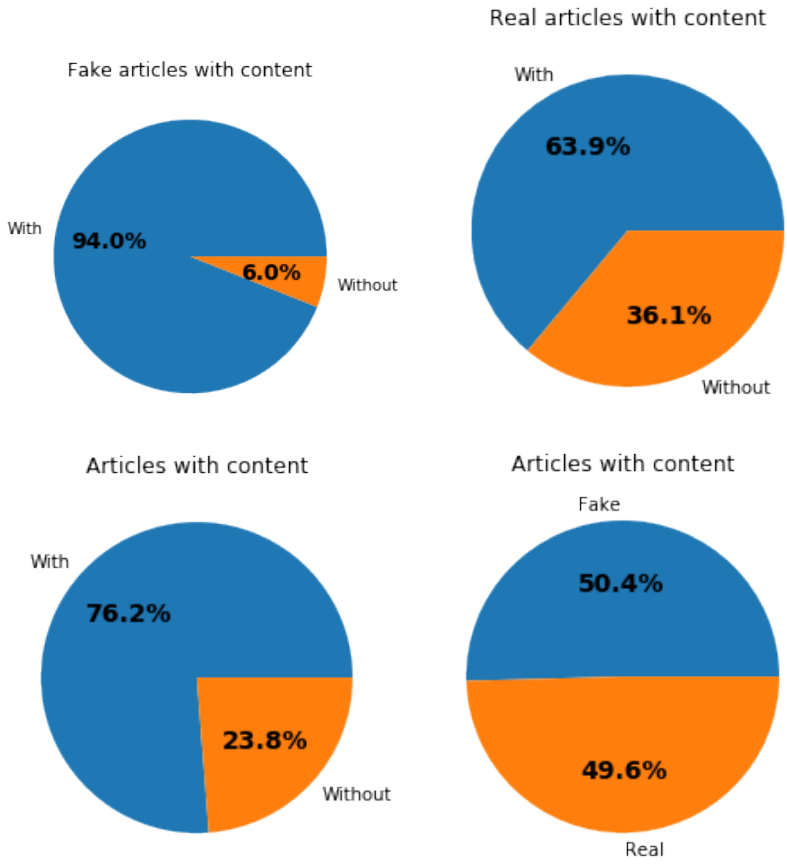
Rysunek 2: Podział Politifact na fake newsy.

Istnieją jednak problemy z pobieraniem artykułów, ponieważ cześć stron nie jest już dostępna w internecie oraz prędkość pobieranie jest niska[5, 6]. Dodatkowo skrypt jest blokowany przez cześć stron z powodu braku zaakceptowanie polityki prywatności lub dlatego że został oznaczony jako szkodzący bot. Statystyki odnośnie ilości artykułów bez treśći zostały umieszczone na obrazku 3

Dla artykułów, które udało nam się pobrać zrobiliśmy rozpoznanie z jakich stron one pochodzą. Sprawdziliśmy że większość stron pojawiła się tylko 1 raz, co widać na obrazku 4). Najpopularniejsze źródła dla fałszywych i prawdziwych artykułów zostały przedstawione odpowiednio w tabelach 1 i 2.

Zauważyliśmy że cześć źródeł ma zarówno fałszywe jak i prawdziwe artykuły. Przygotowaliśmy statystyki, które mówią nam ile jest źródeł mieszanych a ile jest źródeł z jednym typem artykułów, co zostało przedstawione na obrazku 5. Najbardziej popularne źródła mieszane zostały przedstawione w tabeli 3.

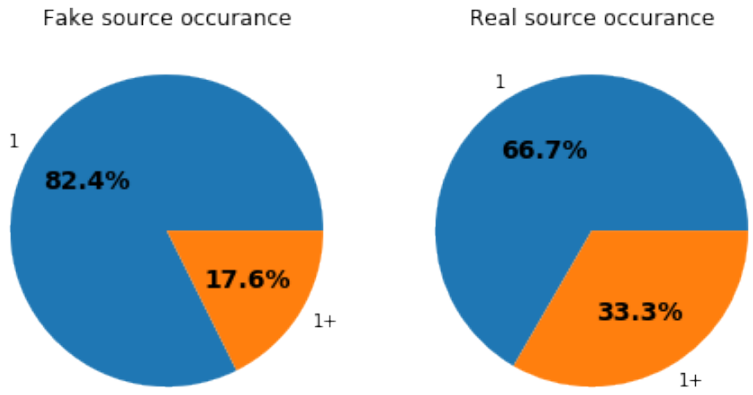
Zauważyliśmy że cześć artykułów ma taki sam tytuł lub treść co inny artykułów. Dane na temat najpopularniejszych tematów i treść unieśliśmy



Rysunek 3: Statystyki odnośnie ilości artykułów dla których nie udało nam się pobrać treści

Tablica 1: Najpopularniejsze źródła fake newsów

URL	Liczba artykułów
https://web.archive.org	169
https://www.facebook.com	6
http://www.react365.com	4
https://yournewswire.com	4
http://thehill.com	4

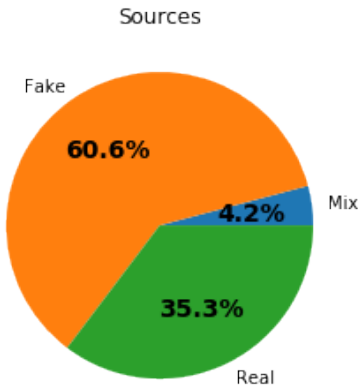


Rysunek 4: Występowanie źródeł

Tablica 2: Najpopularniejsze źródła prawdziwych artykułów

URL	Liczba artykułów
https://web.archive.org	133
http://www.youtube.com	25
http://abcnews.go.com	18
http://www.nytimes.com	15
http://www.cq.com	12

odpowiednio w tabelach 4 i 5. Analizując dane, możemy dojść do wniosku że błędy są spowodowane błędnym skryptem, który został zablokowany np. przez Youtube.



Rysunek 5: Występowanie mieszanych źródeł

Tablica 3: Najpopularniejsze źródeł mieszanych

URL	Liczba fałszywych artykułów	Liczba prawdziwych artykułów	Procent fałszywych artykułów
http://thehill.com	4	2	0.67
https://web.archive.org	169	122	0.58
http://www.businessinsider.com	1	1	0.5
http://nymag.com	1	1	0.5
https://www.washingtonpost.com	3	6	0.33
http://time.com	1	3	0.25
http://www.washingtonpost.com	1	3	0.25
https://www.youtube.com	1	4	0.2
http://www.cnn.com	1	5	0.17
http://www.politifact.com'	2	11	0.15
https://twitter.com	1	6	0.14
https://medium.com	1	8	0.11

Tablica 4: Najpopularniejsze tematy newsów

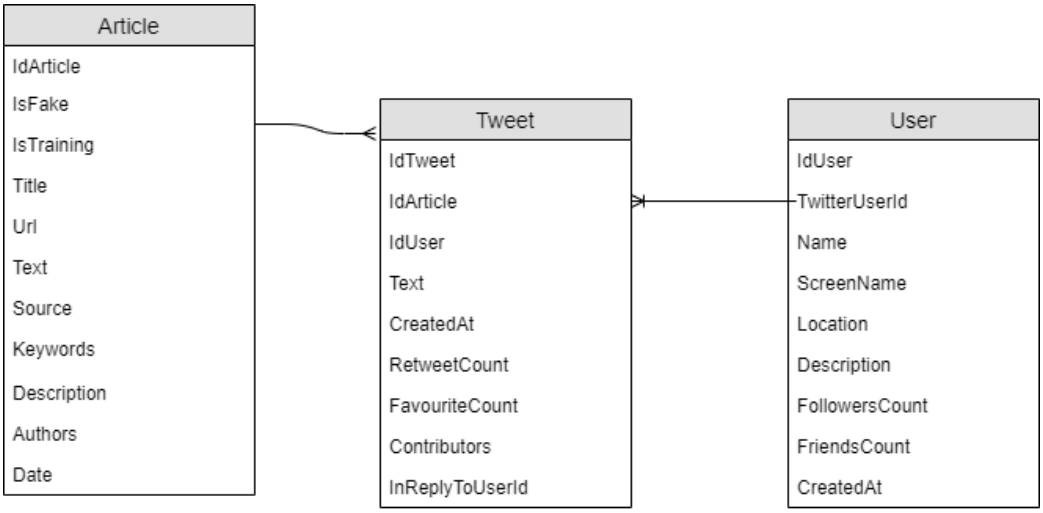
Tytuł	Liczba artykułów
	41
CQ.com	14
- The Washington Post	13
Wiadomości, Pogoda, Outlook, Hotmail, Skype,..	11
YouTube	10
Transcripts	6
Political TV Ad Archive » PolAd	4
Time	4
LexisNexis(R) Publisher	3
MoveOn.org Political Action: 10 things to know..	3
Loading...	2

Tablica 5: Najpopularniejsze treści newsów

Treść	Liczba artykułów
	73
Need help? Contact the CQ Hotline at (80...	14
Please enable cookies on your web browse...	13
About Trendolizer™ Trendolizer™ (patent...	13
Język, w którym oglądasz YouTube, to. Mo...	11
Autoodtwarzanie Jeśli masz włączone auto...	10
Pomiń wszystko Witamy! Na osi czas spęd...	7
Używamy plików cookie, aby pomóc w perso...	5
About Your Privacy on this Site Welcome...	4
Use this guide to help you find the full...	3

5 Architektura bazy danych

W celu przechowania zebranych artykułów oraz tweetów została stworzona relacyjna baza danych. Pierwotna idea zakładała użycie grafowej bazy danych Neo4J, jednak trudności w jej skonfigurowaniu skłoniły do użycia PostgreSQL. Instancja bazy danych została zamieszczona na chmurze AWS jako RDS o rozmiarze db.t2.micro. Schemat bazy ma postać zamieszczony w postaci diagramu. 6



Rysunek 6: Schemat bazy dancyh

Schemat zawiera trzy encje. Pierwsza z nich *Article* ma na celu przechować ściągnięte artykuły. Zawiera takie pola jak:

- IdArticle - id artykułu
- IsFake - flaga czy artykuł jest fałszywy, czy prawdziwy
- IsTraining - artykuły zostały podzielone na zbiór testowy i treningowy, ta flaga mówi do którego dany artykuł należy
- Title - tytuł artykułu
- Url - ścieżka url, pod którym był dostępny artykuł
- Text - pełny tekst artykułu

- Source - źródło, z którego pochodził artykuł
- Keywords - słowa kluczowe opisujące artykuł
- Description - krótki opis
- Authors - autorzy artykułu
- Date - data wydania artykułu

Następną encją jest *User*, która przechowuje użytkowników Twitter'a. Jej pola to:

- IdUser - id użytkownika
- TwitterUserId - id użytkownika używane przez Twittera
- Name - nazwa użytkownika
- ScreenName - nazwa użytkownika wyświetlana w jego opisie
- Location
- Description - tekst widniejący w opisie użytkownika
- FollowersCount - liczba osób obserwujących
- FriendsCount - liczba osób obserwowanych
- CreatedAt - data założenia konta

Ostatnią tabelą jest *Tweet*, która zawiera opublikowane tweety, odnoszące się do danej wiadomości. Jej struktura przedstawia się następująco:

- IdTweet - id tweetu
- IdArticle - id artykułu, do którego się odnosi dany tweet
- IdUser - id użytkownika, który opublikował wpis
- Text - treść wpisu
- CreatedAt - data utworzenia wpisu
- RetweetCount - ilość udostępnień danego wpisu przez inne osoby
- FavouriteCount - ilość polubień wpisu
- Contributors

- InReplyToUserId - wpisy mogą być odpowiedzią na jakiś inny wpis, to pole mówi, któremu użytkownikowi odpowiadamy

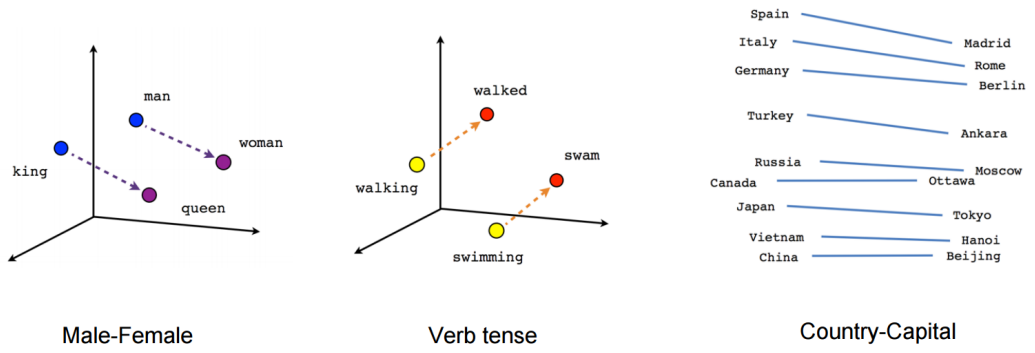
6 Architektura rozwiązania

Z powodu problemów ze zbiorem danych ograniczyliśmy się do używania gotowych plików CSV, gdzie niestety mamy tylko tytuł artykułu, link do artykułu, identyfikator artykułu oraz identyfikatory tweetów odnoszących się do artykułu. Skupiliśmy się na tytule i linku, tym samym na czynnikach, które są wspólne dla wszystkich artykułów oraz są dla nich reprezentatywne. Można by również mieć interpretacje że analizujemy czy dany artykuł jest tzw clickbatiów, czyli zjawiska, gdzie tytuł artykułu ma przyciągnąć naszą uwagę, choć artykuł nie ma treści merytorycznej.

Nasz problem można rozpatrzeć jako problem przetwarzania języka naturalnego. Obecnie trendem w tej dziedzinie są word-embedingi, czyli reprezentowanie słów jako wektor liczbowy. Na takich wektorach potencjalnie możliwe są operacje matematyczne jak dodawanie i odejmowanie, co zostało zaprezentowanie na obrazku 7. Jednak ponieważ wiele słów ma taki sam zapis, ale może oznaczać coś innego powstały specjalne kontekstowe word embedingi. Pozwalają one miedzy innymi zrozumieć kontekst słowa, na podstawie innych słów w zdaniu, które mogą być od niego oddalone. Najbardziej popularną obecnie metodą otrzymania takiej reprezentacji są transformery, którego przykładowa architektura znajduję się na obrazku 8 Rozwiązania takie jak BERT [8] osiągają dużo lepsze wyniki niż n-gramy oraz bags of words i prowadzą do powstania podobnych rozwiązań jak np. RoBERTa, DistilBERT, XLNet czy ALBERT.

W naszym rozwiązaniu użyliśmy biblioteki Flair [7], która jest jedną z najbardziej popularny i najlepszych bibliotek obecnie dostępnych z użyciem transformer.Flair posiada swoje własne modele językowe jak i również umożliwia korzystanie z innych jak np BERT czy ELMO. Dodatkowo Flair ma wbudowane klasyfikatory do tekstu na podstawie zdań, których użyliśmy. Niestety Flair ma też ograniczenia. Jednym z nim jest to że dane treningowe należy mu przekazywać w postaci odpowiednio sformatowanych plików co utrudnia jego użycie oraz wydłuża proces uczenia.

Z powodu dużej popularności modeli parametrycznych jak np. sieci neuro-

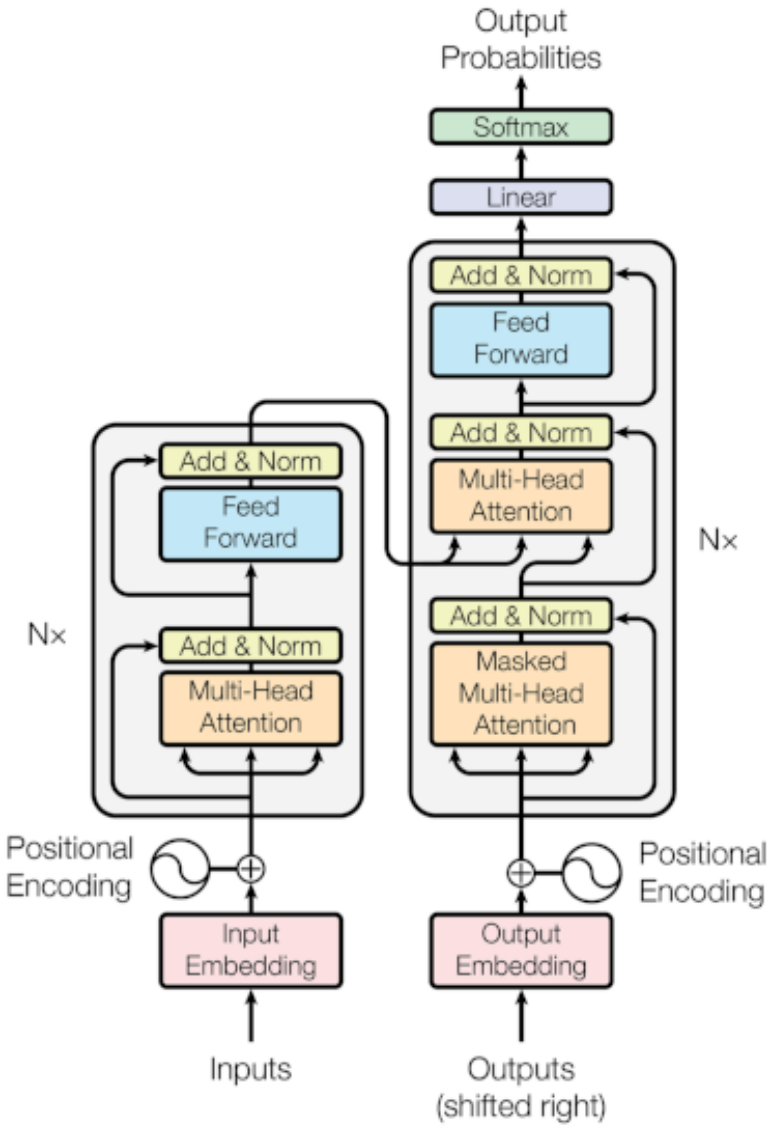


Rysunek 7: Przykład wizualizacji word embedingsów i operacji na nich

nowe powstaję coraz więcej narzędzi, które przyśpieszają proces tworzenia i wybierania takich modeli. Jedną z dzień takich narzędzi jest AUTOML, który pozwala na nie tylko trenowanie modeli w poszukiwaniu optymalnych parametrów modelu, ale również jego hiperparametrów. Przykładami hiperparametrów może być funkcja kosztu lub też cala architektura sieci neuronowej. W tym projekcie użyliśmy narzędzia AutoKeras [9], które oferuję rozwiązanie AUTOML dla popularnej biblioteki Keras. W swojej ofercie ma już gotowe wyuczone modele lub można wytrenować nowy model od nową dla konkretnego zadania. Biblioteka działa na dwóch poziomach. Na wyższym ustala hiperparamterty modelu, który następnie trenuje w niższym poziomie. Gdy model zostanie wyuczony zostaje on oceniony i następnie wybierane są kolejne hiperparamtery, tak aby kolejne modele były coraz lepiej ocenione. Takie rozwiązanie działa o wiele dłużej niż standardowe uczenie modelu, ze względu na ilość trenowanych modeli. Jednak pozwala to uprościć proces dla użytkownika, który nie musi ręcznie porównywać modeli i samemu wymyślać hiperparametrów.

Nasze modele trenowaliśmy na 3 sposoby:

- Title - czyli na samych tytułach
- Url - czyli na samych linkach do artykułów
- Mix - czyli połączenie linku i tytułu po przez konkatenacje ze średnikiem



Rysunek 8: Przykład architektury transformera

7 Ewaluacja

Jako platformę testową używaliśmy Google Colab, która w łatwy sposób umożliwią korzystanie ze środowiska typu Jupyter w internecie. Google Colab jest ponad to darmowy i po za CPU oferuje również możliwość używania

Tablica 6: Porównanie wyników

PolitiFact				
Model	Accuracy	Precision	Recall	F1
Social Article Fusion /S	0.654	0.600	0.789	0.681
Social Article Fusion /A	0.667	0.667	0.579	0.619
Social Article Fusion	0.691	0.638	0.789	0.706
Flair Title	0.816	0.761	0.805	0.782
Flair Url	0.804	0.733	0.860	0.791
Flair Mix	0.759	0.648	0.908	0.756
AutoKeras Mix	0.863	0.811	**0.918**	0.861
HPNF	0.843	0.835	0.851	0.843
UPF	**0.909**	**0.948**	0.864	**0.904**

GPU i TPU. Ponieważ nasze modele były względnie małe postanowiliśmy użyć środowiska z GPU, ponieważ TPU nie przyniosłoby nam korzyści czasowej.

Nasze modele ewaluowaliśmy na podzbiorze Politifact. Zbiór danych podzieliliśmy (względem kolejności w zbiorze danych) na 60% uczących, 20 walidacyjnych oraz 20% testowych, co powinno być zgodne z tym co zrobili twórcy [4]. Na danych treningowych trenowaliśmy swoje modele, a na danych testowych wyliczaliśmy metryki. Dane walidacyjne były używane do ustalenia optymalnych hiperparametrów modeli.

W tabeli 6 umieściliśmy swoje wyniki, wyniki uzyskane przez omawiany przez nas artykułu. Nasze modele to: Flair Title, Flair Url, Flair Mix oraz AutoKeras Mix. Z powodu długości uczenia się modeli AUTOKERAS postanowiliśmy je wytrenować jedynie na sposób Mix.

Wyuczyliśmy również model AutoKeras Mix Overfitted, który jako zbiór walidacyjny dostał zbiór testowy i jak łatwo zauważyć, został on przeuczony do tych danych, choć na ich podstawie były ustalane jedynie hiperparametry sieci. Tym samym pokazuję to jak ważne jest odgraniczenie zbioru treningowe od zbioru testowego.

Analizując na wyniki można dojść do wniosku, że algorytmy, które używają aspektu społecznego, otrzymują lepsze wyniki. Jednak nasze wyniki uzyskane na podstawie wyłącznie tytułów i linków są zadowalające.

8 Podsumowanie

Wykorzystując najnowsze rozwiązania z dziedziny NLP, udało nam się uzyskać lepsze wyniki niż początkowo zakładaliśmy. Tym samym można zauważyć jak bardzo dynamiczną dziedziną jest uczenie maszynowe i eksploracja danych, dlatego należy na bieżąco zaznajamiać się z najnowszymi odkryciami naukowymi.

Ważnym wnioskiem nasuwającym się po tym projekcie jest to aby pisania prac naukowych dobrze opisać swoje badania oraz zbiór danych. Umożliwia to łatwe powtórzenie badania i rzetelne porównanie wyników. Niestety prace naukowe na których bazowaliśmy opisały swoje dane oraz dostarczyły narzędzia jak je pobierać, ale gdyby przygotowali gotową paczkę z danymi to więcej osób mogłoby porównać otrzymane wyniki w rzetelny sposób.

W celu łatwego reprodukowania naszego wyniku, umieściliśmy kod na gitubie[2]. Zostały tam również umieszczone gotowe notebooki, które wystarczy tylko uruchomić na platformie Google Colab.

Literatura

[1] Convolutional neural network for text classification in tensorflow. `https://github.com/dennybritz/cnn-text-classification-tf`.

[2] fake-news-detector. `https://github.com/Kotwic4/fake-news-detector`.

[3] Fakenewsnet - github. `https://github.com/KaiDMML/FakeNewsNet`.

[4] Fakenewsnet podział datasetu. `https://github.com/KaiDMML/FakeNewsNet/issues/1`.

[5] Fakenewsnet problem z pobieraniem danych. `https://github.com/KaiDMML/FakeNewsNet/issues/18`.

[6] Fakenewsnet problem z pobieraniem danych. `https://github.com/KaiDMML/FakeNewsNet/issues/14`.

[7] Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649, 2018.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[9] Haifeng Jin, Qingquan Song, and Xia Hu. Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1946–1956. ACM, 2019.

[10] Kai Shu, Deepak Mahudeswaran, and Huan Liu. Fakenewstracker: a tool for fake news collection, detection, and visualization. *Computational and Mathematical Organization Theory*, 25(1):60–71, 2019.

[11] Kai Shu, Deepak Mahudeswaran, Suhang Wang, Dongwon Lee, and Huan Liu. Fakenewsnet: A data repository with news content, social context and dynamic information for studying fake news on social media. *arXiv preprint arXiv:1809.01286*, 2018.

[12] Kai Shu, Deepak Mahudeswaran, Suhang Wang, and Huan Liu. Hierarchical propagation networks for fake news detection: Investigation and exploitation, 2019.

[13] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. Fake news detection on social media: A data mining perspective. *ACM SIGKDD Explorations Newsletter*, 19(1):22–36, 2017.

[14] Kai Shu, Xinyi Zhou, Suhang Wang, Reza Zafarani, and Huan Liu. The role of user profile for fake news detection, 2019.