Wykrywanie kategorii internetowych artykułów w układzie wielojęzycznym

Katarzyna Bielecka (300165)
Krzysztof Piotrowski (300175)

17 Stycznia 2023

1 Wstęp

Celem projektu jest klasyfikacja artykułów ze względu na perswazyjny charakter tekstu. Koncepcja projektu jest zaczerpnięta z zadania z tegorocznej edycji konkursu SemEval (link do listy zadań konkursowych).

W ramach projektu postanowiliśmy zrealizować pierwszy podpunkt z zadania nr 3. Podpunkt ten polega na zaklasyfikowaniu artykułów w różnych językach pod kątem ich kategorii. Te kategorie to: opinion (opiniotwórcze), reporting (informujące) oraz satire (satyryczne). Każdy artykuł ma przyporządkowaną tylko jedną kategorię. Dokładniejszy opis danych został zamieszczony w sekcji poniżej.

Projekt został napisany w języku Python, używając notatnika Jupyter. Korzystaliśmy także z modułów dla języka Python przydatnych w uczeniu maszynowym i przetwarzaniu języka naturalnego. Były to następujące moduły: pandas, numpy, matplotlib, seaborn, scikit-learn, imblearn, nltk czy hugging face.

2 Zbiór danych

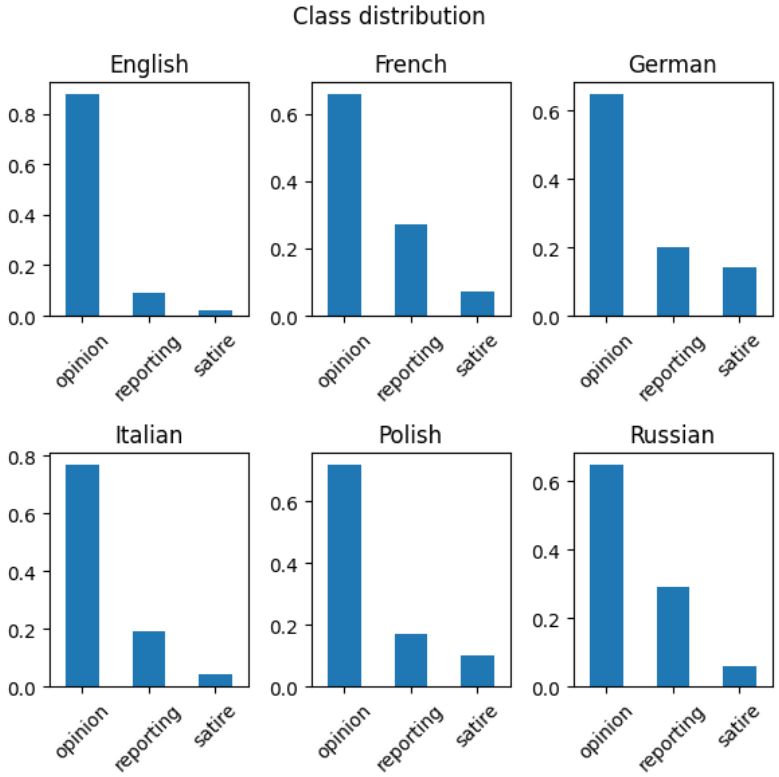
2.1 Charakterystyka tekstów

Artykuły dostępne są w sześciu językach: angielskim, francuskim, niemieckim, włoskim, polskim oraz rosyjskim. Wszystkie teksty zostały napisane między początkiem 2020 roku a połową 2022 roku. Ich tematami są bieżące informacje globalne z danego okresu, np. pandemia Covid-19, zmiany klimatyczne, aborcja, migracja czy wojna w Ukrainie, a także lokalne wydarzenia takie jak wybory. Źródłem artykułów są media popierające wszystkie strony polityczne, zarówno mainstreamowe jak i alternatywne. Większość tekstów została sprawdzona pod względem dezinformacji przez weryfikatorów i ekspertów medialnych.

2.2 Opis danych

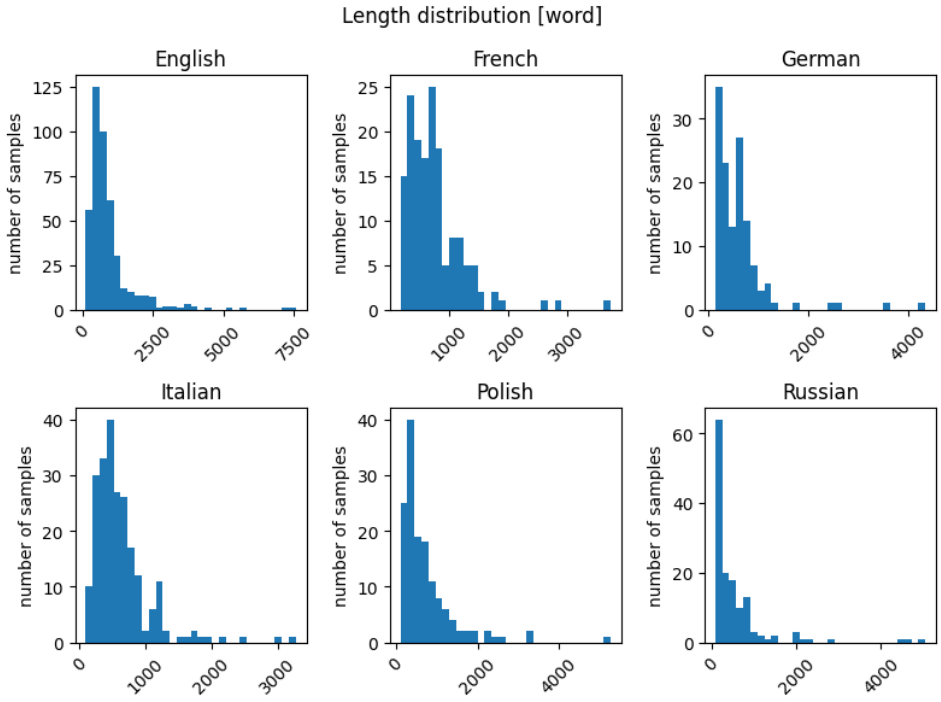
Każdy artykuł znajduje się w osobnym pliku tekstowym i przypisany jest mu numer identyfikacyjny. Artykuły są pogrupowane ze względu na kraj pochodzenia. Pierwszy wiersz zawiera tytuł (jeżeli istnieje), oraz treść artykułu poprzedzoną pustą linią. Teksty podzielone są na trzy kategorie:

- opiniotwórcze - przekazują opinię i odczucia danej osoby. Mają za zadanie przekonać czytelnika do przyjęcia danej pozycji względem jakieś sytuacji albo zmienić jego poglądy.
- informujące - przekazują obiektywne informacje lub też prezentują fakty na dany temat.
- satyryczne - jest to artykuł niezgodny ze stanem faktycznym, który ma za zadanie pokazać i wyśmiać zachowanie lub zjawisko, które jest haniebne lub skorumpowane. Ich celem są publiczne osoby, organizacje lub też pojedyncze wydarzenia. Teksty satyryczne posługują się hiperbolą, absurdem aby szokować, ale jednocześnie nie oszukiwać czytelnika.



Rysunek 1: Rozkład kategorii artykułów dla wszystkich języków

Jak można zauważyć na rysunku 1 znaczną część artykułów w każdym języku stanowią artykuły opiniotwórcze, następnie są informujące, a na końcu satyryczne, które stanowią kilka do kilkunastu procent. Proporcje klas różnią się jednak między językami - na przykład w języku angielskim artykułów satyrycznych czy informujących jest znacznie mniejszy procent niż w języku niemieckim czy francuskim. Taki rozkład klas oznacza, że mamy do czynienia z niezbalansowanym zbiorem danych, co oznacza, że tzw. mało liczne klasy będzie ciężej poprawnie zaklasyfikować. Wobec czego będzie konieczne zastosowanie dodatkowych środków aby temu przeciwdziałać.



Rysunek 2: Rozkład długości artykułów dla wszystkich języków

Artykuły różnią się także długością tekstów. Na Rysunku 2 można zauważyć, że większość artykułów nie przekracza 1000 słów, a najdłuższe z nich nie przekraczają 8000 słów. Mediana długości artykułów w zbiorze treningowym w słowach wynosi 685.

3 Podejście klasyczne

Nasze podejście zaczęliśmy od przygotowania datasetu, usunięciu zbędnych znaków i wektoryzacji tekstów. Następnie przeszliśmy do sprawdzenia czterech. różnych rozwiązań stosowanych przy klasyfikacji tekstów:

- SVM
- Random forest
- Complement Naive Bayes
- Multinomial Naive Bayes

Dla pierwszych dwóch opcji dokonaliśmy regulacji hiperparametrów. Na końcu powiększyliśmy nasz dataset używając metody SMOTE.

3.1 Przetwarzanie danych

Odczytaliśmy dane treningowe i testowe za pomocą dostarczonej przez organizatorów metody ”make_dataframe”, która wczytuje dane z plików tekstowych oraz etykiety i łączy je

w jedną ramkę danych. Następnie sprawdziliśmy rozkład klas oraz rozkład długości artykułów (zamieszczone w poprzednim rozdziale) dla danych z jednego języka - angielskiego. Następnie w kilku etapach przeprowadziliśmy proces przetwarzania tekstu.

W pierwszym kroku przetwarzania tekstu pozbyliśmy się białych znaków takich jak tab, spacja, znak nowej linii oraz znaków przestankowych takich jak m.in. przecinek, kropka, wykrzyknik używając metody str.replace(). Zbędne znaki zostały w ten sposób zamieniony na pusty string lub pojedynczy biały znak.

W kolejnym kroku sprowadziliśmy cały tekst do małych liter używając metody str.lower(). w ten sposób sprawiliśmy by słowa takie jak 'Right' i 'right' były traktowane tak samo.
Następnie usunęliśmy apostrofy, oraz charakterystyczne dla angielskiego pary znaków - 's, np w wyrazie She's. W tym celu ponownie skorzystaliśmy z str.replace().

W następnym etapie wykluczyliśmy z artykułów słowa należące do tzw. stop listy, czyli spójniki, zaimki oraz słowa występujące często, które nie są charakterystyczne dla pojedynczego tekstu. Przykłady słów ze stop listy dla języka angielskiego to: 'me', 'you', 'which', 'at', 'the'. Skorzystaliśmy z gotowej stop listy z modułu nltk.
Opisane słowa usunęliśmy w następujący sposób: Dla każdego wiersza z kolumny z tekstami sprawdziliśmy każde słowo - czy należy do stop listy czy nie. Jeśli do niej nie należy, to było ono dodawane do tablicy tworzącej nowy, oczyszczony ze 'stopwordów' tekst.
Jednocześnie, w tej samej pętli przeprowadziliśmy lemmatyzację, czyli proces sprowadzenia każdego słowa do jego podstawowej formy. Np dla czasowników taką formą jest bezokolicznik. Do lemmatyzacji użyliśmy obiektu klasy WordNetLemmatizer, z modułu nltk. Wracając do opisywanej pętli - jeśli dane słowo nie należy do stop listy, to najpierw było sprowadzane do podstawowej formy a potem dopiero dodawane do nowego tekstu. W ten sposób w jednej pętli zawarliśmy dwa etapy.
W kolejnym kroku ze zbiór trenującego wydzieliśmy zbiór treningowy i testowy, używając metody train_test_split z modułu scikit-learn. Za wartość parametru test_size przyjęliśmy 0.3, czyli w miarę dużą, ze względu na małoliczne klasy reporting i satire.

Do reprezentacji tekstów postanowiliśmy użyć n-gramów, a dokładniej unigramów i bigramów, ze względu na niewielkie długości artykułów. W drodze eksperymentów sprawdzaliśmy również trigramy ale ich zastosowanie razem z uni i bigramami przyniosło mniej zadowalające rezultaty. Wektoryzację tekstów przeprowadziliśmy za pomocą gotowego elementu TfidfVectorizer z modułu scikit-learn.

3.2 Regulowanie hiperparametrów

W przypadku modelu SVM sprawdzamy 4 parametry: penalty, kernel, gamma i degree. Natomiast dla modelu Random forest sprawdzane są: estimators, max features, max depth, min samples split, min samples leaf oraz bootstrap. Na początku wykonuje się algorytm *Randomized search*, który otrzymuje przedział wartości dla poszczególnych parametrów i w procesie dopasowywania wskazuje parametry dające najlepszą dokładność. Nastepny jest *Grid Search*, który dostaje zawężony zakres wartości parametrów w okolicy wartości wskazanej przez poprzednią metodę. Finalnie zapisywane są najlepsze parametry i przekazane do trenowania.

3.3 Powiększanie zbioru przy użyciu metody SMOTE

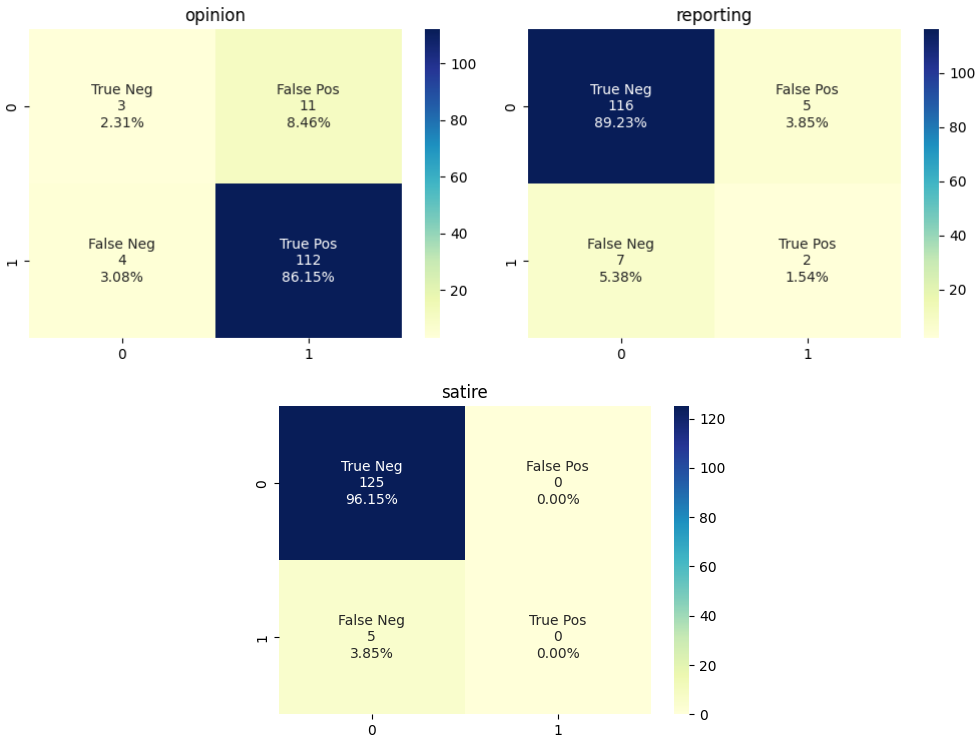
Przy pierwszych próbach trenowania, żaden z modeli nie był wstanie wskazać artykułów innych niż opiniotwórcze. Wynika to z niezbalansowania zbioru danych, w którym artykuły inne niż opiniotwórcze stanowią zaledwie 12%, w tym satyry stanowią tylko 2%. Aby rozwiązać ten problem użyliśmy metody SMOTE, która generuje syntetyczne dane dla klas mniejszości. Polega to na kopiowaniu istniejących danych i wprowadzaniu w nich niewielkich zmian. Niestety, wielowymiarowość naszych danych sprawiło, że osiągnięte wyniki były obiecujące dla danych treningowych, ale nie poprawiły się dla danych testowych.

3.4 Wyniki na zbiorze testowym

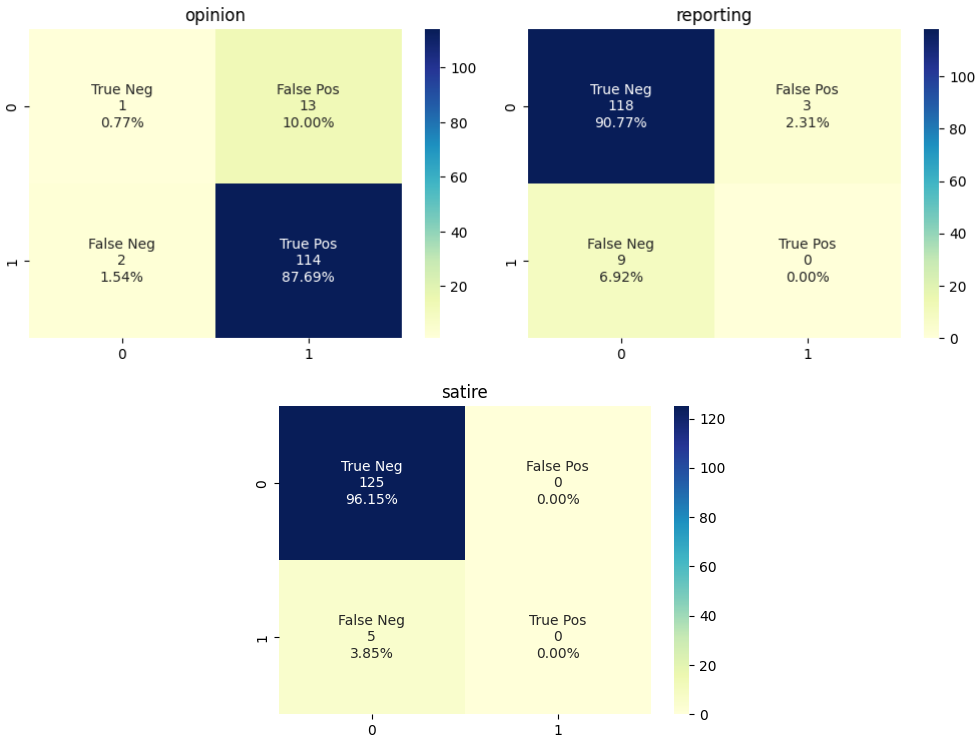
W tabeli 1 przedstawiono metryki dla wszystkich modeli na zbiorze testowym. Wszystkie modele, oprócz Complement Naive Bayes, uzyskały wynik dokładności w przedziale 88-89%. Dwóm modelom udało się rozpoznać przynajmniej jeden artykuł informujący: SVM i Complement Naive Bayes, a ten drugi rozpoznał również satyry. Nie są to wyniki zadowalające, ale jest to dobry punkt wyjścia do przyszłego ulepszenia hiperparametrów.

	SVM	Random forest	Comp. Naive Bayes	Mult. Naive Bayes
Dokładność	0.88	0.88	0.88	0.71
Precyzja	0.40	0.30	0.46	0.30
Zupełność	0.40	0.33	0.46	0.33
Miara F1	0.40	0.31	0.42	0.31

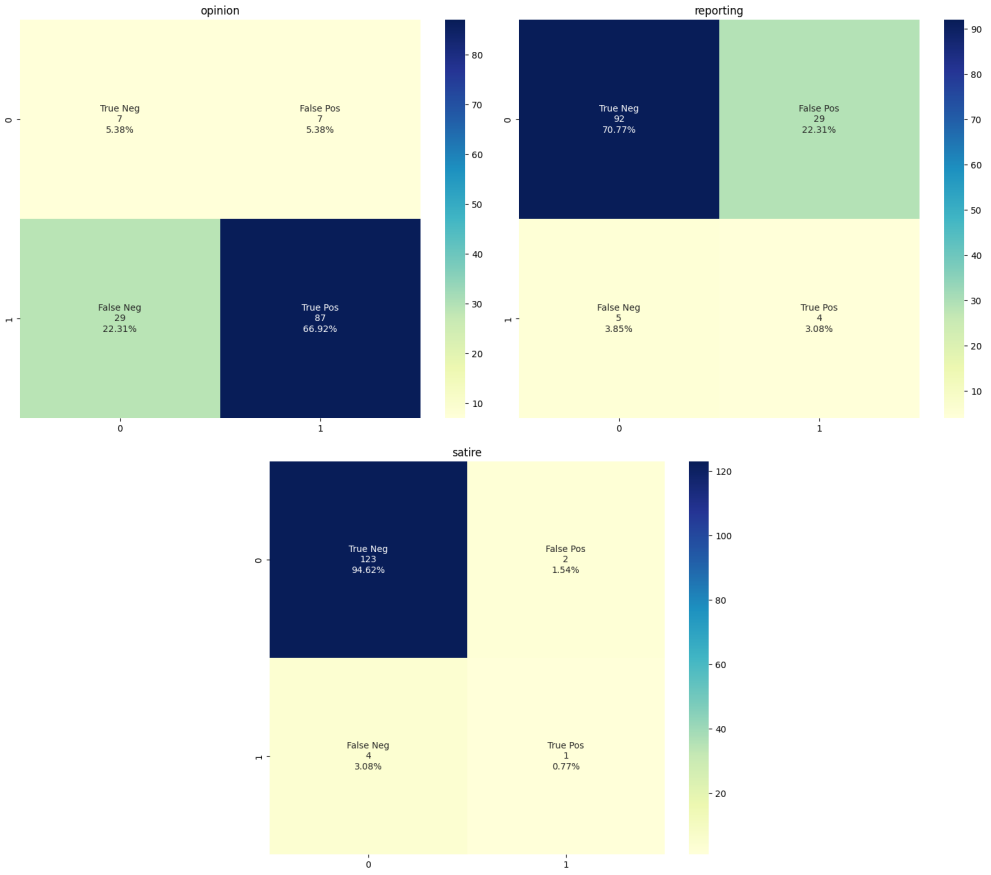
Tabela 1: Wyniki dla poszczególnych modeli



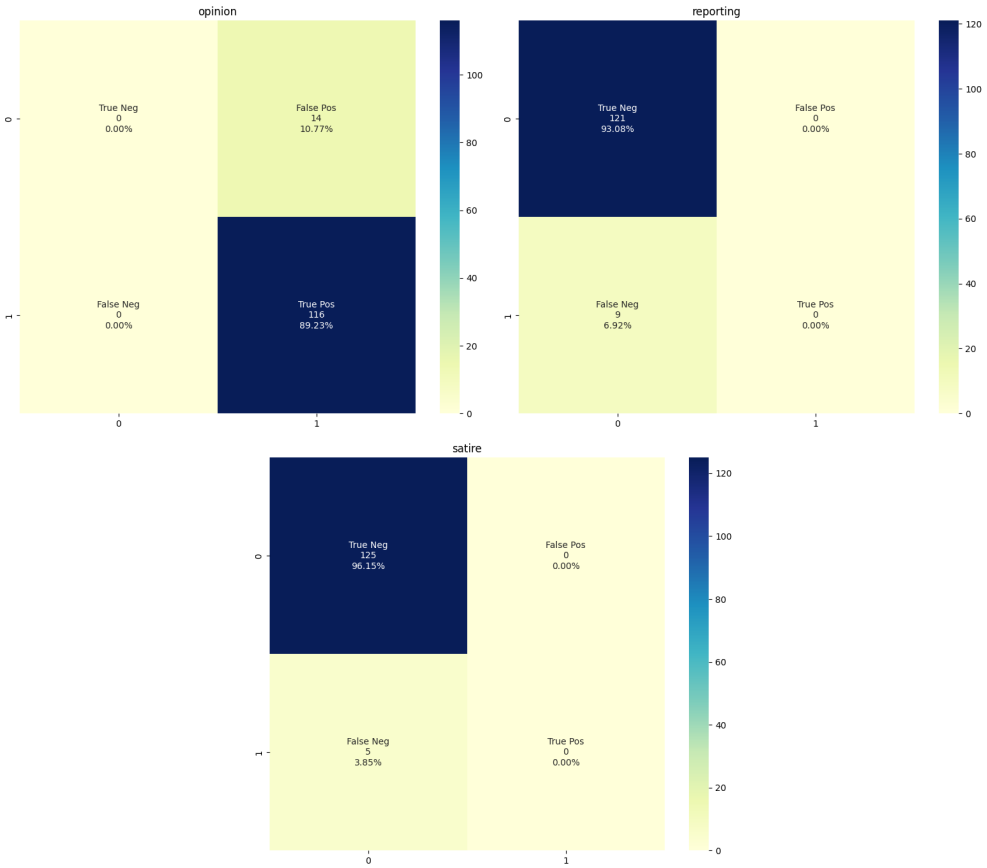
Rysunek 3: Macierze błędów dla modelu SVM



Rysunek 4: Macierze błędów dla modelu Random Forest



Rysunek 5: Macierze błędów dla modelu Complement Naive Bayes



Rysunek 6: Macierze błędów dla modelu Multinomial Naive Bayes

4 Podejście nowoczesne

W następnym podejściu użyliśmy gotowych, wytrenowanych modeli dostępnych na platformie Hugging Face, które przystosowane są do klasyfikacji tekstów wielojęzycznych. Dokonaliśmy tuningu modeli pod nasze zadanie, a także dokonaliśmy oversamplingu niezbilansowanych klas.

4.1 Użyte modele

Zdecydowaliśmy się na użycie modeli typu BERT (*Bidirectional Encoder Representations from Transformers*). Modele te, oparte na transformatorach, polegają na tym, że każdy element wyjściowy jest połączony z każdym elementem wejściowym, a wagi między nimi są obliczane dynamicznie na podstawie ich połączenia. W szczególności skupiliśmy się na dwóch odmianach tego modelu: *bert-multilingual-uncased*, który został przetrenowany na 102 językach opierając się o teksty z Wikipedii oraz *XlmRoBERTabase*, który został wytrenowany na 100 językach opierając się o 2.5 TB przefiltrowanych tekstów z CommonCrawl. Obie opcje mają automatyczne rozpoznawanie języka, dzięki czemu nie trzeba dostarczać informacji o aktualnie uczącym się języku. W obu przypadkach dokonaliśmy regulacji hiperparametrów poprzez wbudowane narzędzie w Hugging Face o nazwie *hyperparameter_search*.

4.2 Wstępne wyniki

Modele klasyczne trenowaliśmy i testowaliśmy tylko i wyłącznie na języku angielskim. W przeciwieństwie do tego, modele BERT wytrenowaliśmy i przetestowaliśmy na wszystkich 6 językach.
Jak pokazuje tabelka 2 model xlm-RoBERTa-base nie osiągnął satysfakcjonujących wyników. Wartość miary F1 wyniosła 28%, co wynika z tego,że nie był w stanie rozpoznać artykułów informujących i satyrycznych.
Z drugiej strony zaobserwowaliśmy znaczącą poprawę wyników w przypadku modelu BERT-base-multilingual. Model ten osiągnął wartość miary F1 na poziomie 58%, co oznacza rozpoznanie artykułów inne niż opiniotwórcze. Jest to wynik znacznie lepszy, niż jakikolwiek w podejściu klasycznym.

	xlm-RoBERTa	BERT-multilingual
Dokładność	0.76	0.79
Precyzja	0.33	0.54
Zupełność	0.25	0.68
Miara F1	0.28	0.58

Tabela 2: Wstępne wyniki dla modelów BERT

4.3 Oversampling danych

Aby uzyskać lepsze wyniki zdecydowaliśmy się na oversampling danych metodą podwójnego tłumaczenia. Do tego celu wykorzystaliśmy moduł dla języka Python *googletrans*. Na samym początku artykuł był tłumaczony na inny język, preferowany był język azjatycki z uwagi na większą różnicę względem oryginalnego tekstu, a następnie był tłumaczony z powrotem na

język oryginalny. Dzięki temu artykuł zachowywały swój pierwotny sens, przy jednoczesnych zmianach szyku zdania i słów, co pozwoliło wykorzystać je ponownie w procesie uczenia. W ten sposób powielone zostały artykuły z satyryczne i informujące. Zdecydowaliśmy się tłumaczyć na dwa różne języki - japoński i chiński, co pozwoliło potroić ilość artykułów w obu ze wspomnianych kategorii.

	xlm-RoBERTa	BERT-multilingual
Dokładność	0.57	0.79
Precyzja	0.39	0.60
Zupełność	0.35	0.65
Miara F1	0.35	0.62

Tabela 3: Wyniki po oversamplingu danych

Jak można zauważyć w tabelce 3, dla obu modeli zanotowaliśmy poprawę miary F1. Jest to odpowiednio 7% dla modelu xlm-RoBERTa i 4% dla BERT-base-multilingual. Jednakże dokładność modelu xlm-RoBERTa spadła do wartości 57%.

4.4 Tuning modelu i zapobieganie przeuczeniu

Po powiększeniu zbioru zaczęliśmy dostrajać hiperparametry naszego modelu używając metody *hyperparameter_search*. Po przeprowadzeniu dziesięciu prób dobraliśmy odpowiednie wartości dla learning rate, weight decay i batch size, co pozwoliło uzyskać lepsze wyniki metryk.

	xlm-RoBERTa	BERT-multilingual
Dokładność	0.75	0.74
Precyzja	0.25	0.76
Zupełność	0.33	0.56
Miara F1	0.28	0.60

Tabela 4: Wyniki po oversamplingu danych + tuningu modelu

W tabeli 4 przedstawiono wyniki uzyskane po trenowaniu na modelu z wyregulowanymi parametrami. Metryki osiągnęły niższą wartość od tych uzyskanych na samym oversamplingu. Zauważyliśmy również zjawisko przeuczenia modelu, gdzie wartość funkcji kosztu dla zbioru walidacyjnego zaczęła rosnąć względem spadku funkcji kosztu dla zbioru treningowego. Aby to zniwelować, zastosowaliśmy dropout o prawdopodobieństwie 20-45%, który wyhamował wzrost funkcji kosztu. Jego wpływ można zaobserwować w tabeli 6.

	xlm-RoBERTa	BERT-multilingual
Dokładność	0.75	0.68
Precyzja	0.25	0.67
Zupełność	0.33	0.63
Miara F1	0.28	0.60

Tabela 5: Wyniki po oversamplingu danych + tuningu modelu + dropout

4.5 Wyniki dla poszczególnych języków

Poniższe tabele przedstawiają wyniki dla poszczególnych języków przy najlepszej konfiguracji parametrów i technik - model BERT multilingual, z tuningiem, oversamplingiem i domyślnym dropoutem 0.1.

	angielski	francuski	polski	włoski	rosyjski	niemiecki
Dokładność	0.77	0.88	0.82	0.85	0.66	0.70
Precyzja	0.36	0.91	0.68	0.56	0.53	0.72
Zupełność	0.36	0.76	0.83	0.56	0.53	0.52
Miara F1	0.36	0.81	0.74	0.56	0.53	0.57

Tabela 6: Wyniki dla wszystkich języków

5 Wnioski

Analizując wyniki klasycznych modeli uczenia maszynowego można śmiało stwierdzić, że nie były one wystarczające dla przedstawionego w zadaniu zbioru danych. Natomiast po zastosowaniu przez nas nowszych rozwiązań stosowanych w NLP osiągnęliśmy znaczną poprawę wyników - lepsze modele, oversampling przez tłumaczenie.
Mimo to, wyniki, szczególnie dla najmniej zbalansowanych języków, takich jak angielski mogłyby być lepsze. Możliwe, że inne, niesprawdzone przez nas modele pozwoliłyby na poprawę klasyfikacji artykułów. Inną rzeczą, wartą sprawdzenia byłyby inne metody stosowane dla zbiorów niezbalansowanych jak np. data augmentation.

References

[1] Alexis Conneau et al. "Unsupervised Cross-lingual Representation Learning at Scale". In: *CoRR* abs/1911.02116 (2019). arXiv: 1911.02116. URL: http://arxiv.org/abs/1911.02116.

[2] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

[3] Yogesh Kothiya. "How I handled imbalanced text data". In: (2019). URL: https://towardsdatascience.com/how-i-handled-imbalanced-text-data-ba9b757ab1d8.

[4] Edward Ma. "Data Augmentation library for text". In: (2019). URL: https://towardsdatascience.com/data-augmentation-library-for-text-9661736b13ff.

[5] Giovanni Da San Martino et al. *Detecting the Genre, the Framing, and the Persuasion Techniques in Online News in a Multi-lingual Setup*. 2022. URL: https://propaganda.math.unipd.it/semeval2023task3/.

[6] *Text classification*. Google. 2022. URL: https://developers.google.com/machine-learning/guides/text-classification/.

[7] Alina Zhang. "Text Data AugmentationWith Google Translate in NLP Projects: How to Solve the Lack of Label Data Problem". In: (2022). URL: https://alina-li-zhang.medium.com/text-data-augmentation-with-google-translate-in-nlp-projects-how-to-solve-the-lack-of-label-data-2873ec752d0c.