[NLP 25L] The Food Hazard Detection Challenge
Dokumentacja końcowa projektu

Michał Kaczmarczyk
Piotr Kitłowski
Hubert Podlewski

Maj 2025

1 Wstęp

W ramach projektu podjęliśmy się wyzwania dziewiątego przedstawionego na tegorocznych (2025) warsztatach SemEval. Zadanie to znajduje się w kategorii „Fact Checking and Knowledge Verification" i koncentruje się na automatycznej klasyfikacji zagrożeń związanych z bezpieczeństwem żywności.

Zgodnie z opisem organizatorów, celem zadania jest utworzenie wyjaśnialnych systemów klasyfikacji tytułów raportów o wypadkach gastronomicznych. Wyjaśnialność systemu ma tu grać kluczową rolę, umożliwiając człowiekowi szybką weryfikację poprawności klasyfikacji.

Bezpieczeństwo żywności jest kluczowym zagadnieniem zdrowia publicznego – każdego roku na świecie odnotowuje się miliony przypadków zatruć i chorób wywołanych przez skażoną żywność. Tradycyjne metody monitorowania takich zagrożeń często opierają się na ręcznej analizie raportów przez ekspertów, co jest czasochłonne i podatne na błędy ludzkie.

Projekt ten stanowi kompleksowe podejście do automatyzacji tego procesu, wykorzystując różnorodne techniki uczenia maszynowego od klasycznych metod do nowoczesnych algorytmów ensemble.

2 Charakterystyka zadania i uzasadnienie wyboru ścieżki

2.1 Podzadania

W ramach projektu skupiliśmy się na pierwszym z dwóch podzadań konkursu:

- **ST1** - tekstowa klasyfikacja przewidująca kategorię zagrożenia (10 klas) i kategorię produktu (22 klasy).
- **ST2** - klasyfikacja przewidująca konkretne zagrożenie (128 klas) i konkretne produkty (1142 klasy) - nie implementowano.

2.2 Dane i przykłady

Zbiór danych składa się z 6,644 ręcznie oznakowanych raportów o odwołaniach produktów żywnościowych z oficjalnych stron agencji żywnościowych (np. FDA, FSA). Dane zostały podzielone

na:

- **Zbiór treningowy**: 5,082 próbek
- **Zbiór walidacyjny**: 565 próbek
- **Zbiór testowy**: 997 próbek

2.2.1 Przykłady danych

Poniższa tabela przedstawia charakterystyczne przykłady z datasetu:

Tabela 1: Przykłady danych z datasetu SemEval 2025 Task 9

Tytuł	Hazard Category	Product Category	Hazard	Product
"Oregon Lox Company Recalls Wild Cold Smoked Keta Salmon Lox Because of Possible Health Risk"	biological	seafood	listeria monocytogenes	salmon
"Lidl recalls HealthyCo Proteinella Smooth Hazelnut and Cocoa Spread because allergenic ingredients are not declared in English"	allergens	nuts, nut products and seeds	milk and products thereof	hazelnuts
"pilgrim's pride corp. recalls poultry products due to possible foreign matter contamination"	foreign bodies	meat, egg and dairy products	foreign bodies	chicken based products
"AG Barr is recalling 750ml glass bottles due to a manufacturing fault which may cause bottle caps to pop off unexpectedly"	packaging defect	non-alcoholic beverages	bulging packaging	soda drink
"Natures Aid is Withdrawing All Batches of its Turmeric due to High Levels of Curcumin"	food additives and flavourings	herbs and spices	other	turmeric powder (curcuma)

Przykłady pokazują różnorodność problemów bezpieczeństwa żywnościowego:

- **Biological hazards**: Bakterie (Listeria, Salmonella)
- **Allergens**: Niedozwolone składniki alergiczne
- **Foreign bodies**: Zanieczyszczenia fizyczne (plastik, metal)
- **Packaging defects**: Wady opakowań
- **Chemical issues**: Nadmierne stężenia substancji

2.3 Niezrównoważenie klas

Jednym z głównych wyzwań projektu było niezrównoważenie klas:

- **Kategorie zagrożeń**: stosunek najczęstszej do najrzadszej klasy wynosi 618:1
- **Kategorie produktów**: stosunek najczęstszej do najrzadszej klasy wynosi 287:1

2.4 Metryka oceny

Do oceny modelu wykorzystano oficjalną metrykę z konkursu SemEval:

```
def compute_score(hazards_true, products_true,
                  hazards_pred, products_pred):
    f1_hazards = f1_score(hazards_true, hazards_pred, average='macro')

    correct_hazard_mask = hazards_pred == hazards_true
    if sum(correct_hazard_mask) > 0:
        f1_products = f1_score(
            products_true[correct_hazard_mask],
            products_pred[correct_hazard_mask],
            average='macro'
        )
    else:
        f1_products = 0.0

    return (f1_hazards + f1_products) / 2
```

Metryka ta priorytetowo traktuje poprawność klasyfikacji zagrożeń - ocena produktów jest liczona tylko dla próbek, gdzie zagrożenie zostało poprawnie zidentyfikowane. Dalej podane wyniki odnoszą się do tej właśnie metryki, chyba że napisano inaczej.

3 Metodyka i uzasadnienie wyborów technicznych

3.1 Przegląd rozważanych alternatyw

Przed implementacją rozważyliśmy kilka alternatywnych podejść:

3.1.1 Zaimplementowane

1. **Zaawansowane transformatory**: BERT-base, BERT-large i Qwen3
    - *Powód*: Używaliśmy dwóch kart graficznych typu NVIDIA GeForce RTX 2080 Ti
    - *Czasochłonność*: Fine-tuning BERTa zajął niecałe 30 minut, BERT-large około 1 godziny, a Qwen3 kilka godzin na 20 epok

3.1.2 Nie zaimplementowane

1. **Syntetyczna Augmentacja Danych**: Parafrazy wygenerowane przez LLM (kluczowa technika zwycięzców)
2. **Wielozadaniowy proces uczenia**: Połączenie zadań ST1, ST2 i optymalizacji
    - *Powód rezygnacji*: Złożoność implementacyjna w świetle ograniczeń czasowych
    - *Literatura*: Wiodące rozwiązania pokazują, że podejścia jednozadaniowe często potrafią być lepsze

3.1.3 Wybrane podejście: Klasycze algorytmy uczenia maszynowego z optymalizacją

Zdecydowaliśmy się na systematyczną optymalizację tradycyjnych metod ML z następujących przyczyn:

- **Praktyczność**: Możliwość implementacji w ograniczonym czasie i na słabym sprzęcie
- **Baseline beating strategy**: Skupienie na pokonaniu BERT baseline (0.667) jako realistycznym celu
- **Koszt**: Tradycyjne algorytmy uczenia maszynowego wymagają mniej mocy obliczeniowej
- **Wyjaśnialność**: Prostota i zrozumiałe przez ludzi matematyczne tło klasycznych modeli sprawia, że jesteśmy realistycznie w stanie przeanalizować, co wpłynęło na podjęte przez nie decyzje

3.2 Modele bazowe

3.2.1 Klasyfikator większości

Najprostszy model referencyjny, który zawsze przewiduje najczęściej występujące klasy zagrożenia i produktu w danych treningowych. Służy jako punkt odniesienia dla bardziej zaawansowanych metod.

3.2.2 TF-IDF + Regresja Logistyczna

Klasyczne podejście oparte na reprezentacji bag-of-words:

- **Wektoryzacja**: TF-IDF z 1-gramami i 2-gramami
- **Przygotowanie danych (ang. preprocessing)**: łączenie tytułu i treści, normalizacja tekstu
- **Klasyfikacja**: dwa oddzielne modele regresji logistycznej
- **Wagi klas**: wykorzystane, by zmniejszyć negatywny wpływ niezrównoważenia danych

3.3 Systematyczna optymalizacja TF-IDF

Przeprowadziliśmy grid search po kluczowych parametrach TF-IDF:

Tabela 2: Wyniki optymalizacji parametrów TF-IDF na zbiorze walidacyjnym

Konfiguracja	max_features	ngram_range	Hazard F1	Uwagi
original	10,000	(1,2)	0.6892	Baseline
more_features	**15,000**	**(1,2)**	**0.7017**	**Najlepszy**
trigrams	10,000	(1,3)	0.6726	Gorszy
less_restrictive	10,000	(1,2)	0.6918	Nieznacznie lepszy

Wniosek: Zwiększenie liczby atrybutów z 10 tys. do 15 tys. dało poprawę, ale zastosowanie trigramów pogorszyło wyniki - prawdopodobnie przez wprowadzenie szumu.

3.4 XGBoost - Uzasadnienie wyboru i implementacja

3.4.1 Dlaczego XGBoost?

Wybór XGBoost był oparty na następujących przesłankach:

1. **Praca z rozproszonymi (ang. sparse) danymi**: XGBoost doskonale radzi sobie z wielowymiarowymi macierzami rzadkimi (TF-IDF)
2. **Nierówny rozkład klas**: Wbudowane mechanizmy sample weighting
3. **Regularyzacja**: Wbudowana regularyzacja L1/L2 zapobiega przeuczeniu
4. **Jakość**: XGBoost często zwraca wyniki porównywalne z podstawowym BERT-em
5. **Wyjaśnialność**: Możliwy poprzez feature importance analysis

3.4.2 Implementacja i wyzwania techniczne

Wykorzystanie XGBoost-a z optymalizacją hiperparametrów wymagał rozwiązania kilku problemów:

```
# Optimized XGBoost configuration
XGBClassifier(
    n_estimators=200,       # Wiecej drzew
    max_depth=8,            # Glebsze drzewa
    learning_rate=0.05,     # Wolniejsze uczenie
    subsample=0.8,          # Zapobieganie przeuczeniu
    colsample_bytree=0.8,   # Losowe wybieranie cech
    random_state=42
)
```

Kluczowe wyzwania techniczne:

- **Label encoding**: XGBoost wymaga liczbowych etykiet - implementacja bezpiecznego kodowania dla nienapotkanych etykiet
- **Macierze rzadkie**: Konwersja rzadkich macierzy TF-IDF do formatu gęstego (kosztowne pamięciowo)
- **Wagi**: Adaptacja wag klas (ang. class weights) z frameworku scikit-learn dla API XGBoost-a
- **Spójność między zbiorami**: Obsługa etykiet występujących tylko w zbiorze treningowym lub ewaluacyjnym

3.5 Eksperymenty z Inżynierią Cech (ang. Feature Engineering)

Próbowaliśmy wzbogacić model o 28 dodatkowych cech:

- **Statystyki tekstowe**: długość tekstu, liczba słów, proporcje
- **Dane czasowe**: rok, miesiąc, dzień, sezon

- **Kodowanie geograficzne**: najważniejsze kraje
- **Cechy semantyczne**: obecność słów kluczowych (recall, contamination, allergen)

Wynik: Pogorszenie jakości z 0.5978 do 0.5126 (-0.085 punktów)

Ten eksperyment pokazał, że **„więcej cech" nie zawsze oznacza lepsze wyniki** - nadmiar cech może wprowadzać szum i utrudniać uczenie modelu.

3.6 Smart Ensemble

Zamiast prostego głosowania większościowego, zaimplementowaliśmy:

- **Głosowanie ważone**: Wagi oparte na jakości na zbiorze walidacyjnym
- **Wybór modeli**: Wybrane zostały automatycznie najlepsze modele, XGBoost otrzymał wagę 0.7, LogReg 0.3

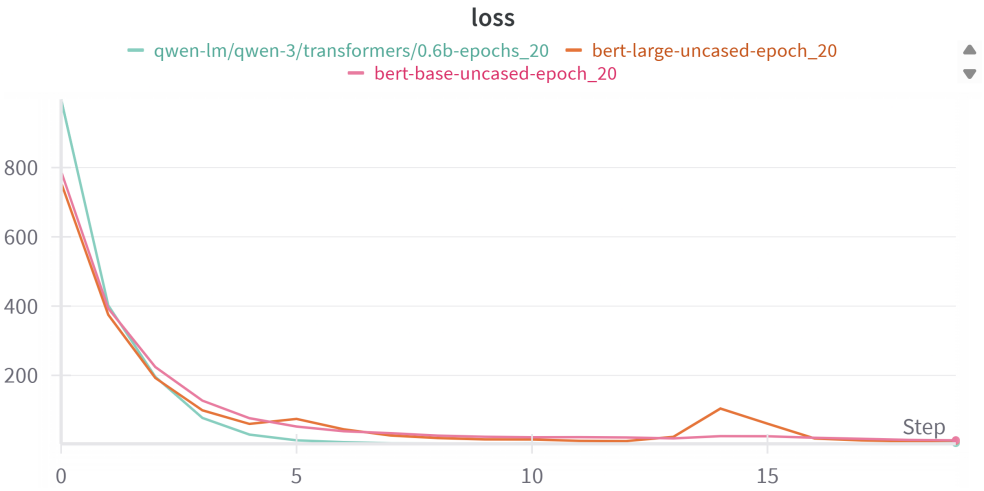
3.7 Dostrajanie (ang. finetuning) Modeli Językowych - BERT i Qwen

Używaliśmy gotowych, wytrenowanych modeli - **BERT base**, **BERT large** i **Qwen3 (0.6b)**, które następnie dodatkowo trenujemy na naszych danych. Dla wszystkich wykorzystanych modeli używaliśmy podstawowych parametrów, jak np.:

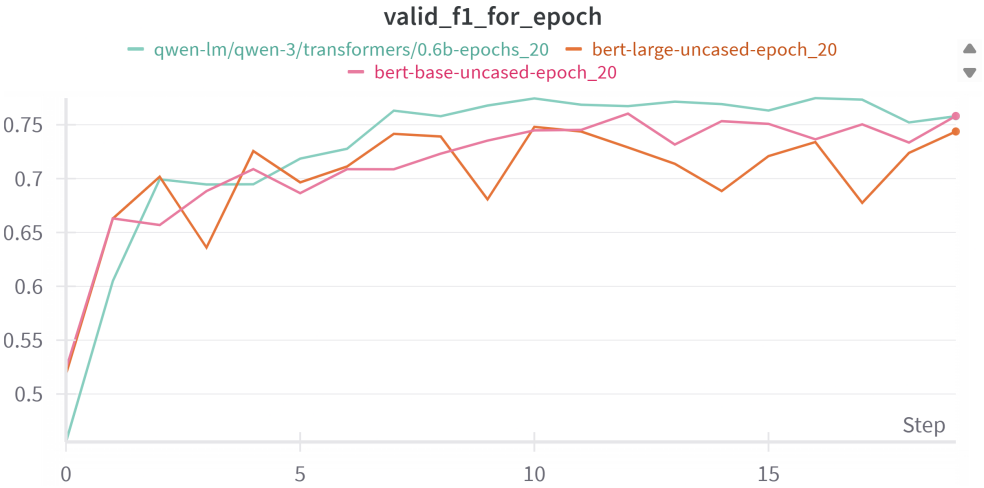
- learning_rate = (1e-5, 2e-5)
- epochs = 20
- batch_size = 16

4 Wyniki końcowe

Rysunek 1: Wykres z wandb pokazuje wartość funkcji straty względem epoki



Rysunek 2: Wykres wandb pokazuje wartości metryki F1 na zbiorze walidacyjnym względem epoki



Rysunek 3: Graficzne porównanie z wandb - metryka F1 dla danych walidacyjnych i testowych

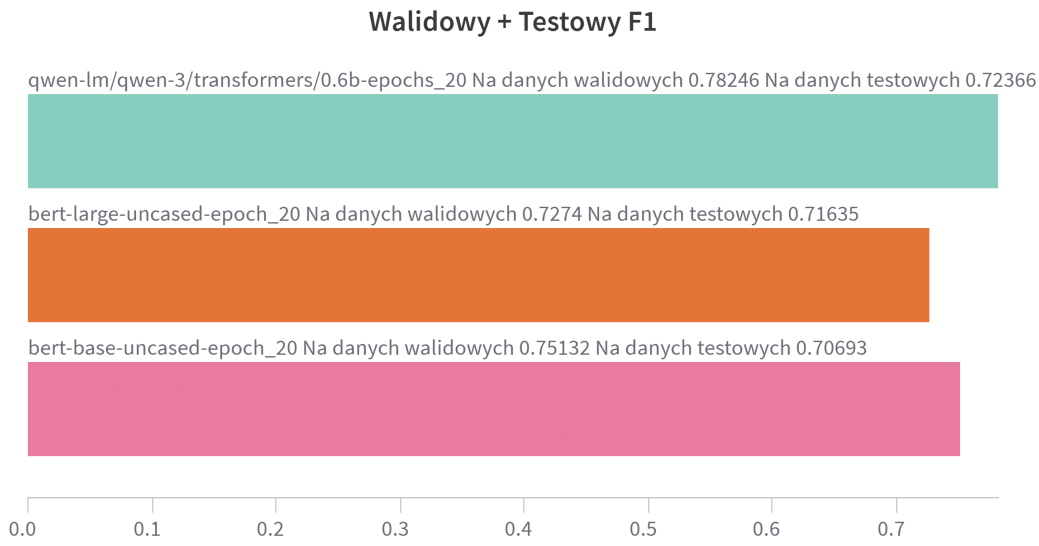


Tabela 3: Porównanie wszystkich zaimplementowanych metod

Podejście	Valid F1	Test F1
Majority Classifier	0.035	0.035
TF-IDF + LogReg (original)	0.611	0.598
Enhanced Features	0.551	0.513
Optimized TF-IDF	0.623	0.616
XGBoost (Enhanced)	0.712	0.666
Smart Ensemble	0.645	0.666
BERT baseline	0.751	0.707
BERT large	0.727	0.0716
Qwen3 0.6b	**0.782**	**0.724**

4.1 Kluczowe osiągnięcia

- **Pokonanie baseline'u BERT-a**: Nasz najlepszy model (dostrojony Qwen) osiągnął wynik 0.724, czyli większy niż oficjalny baseline BERT-a, który wynosił 0.667
- **Znacząca poprawa**: +21.1% względem oryginalnego TF-IDF baseline
- **Praktyczne znaczenie**: Transformatory są przyszłością AI. Dostrojenie gotowego LLM-a jest proste, a uzyskane wyniki są lepsze w porównaniu z klasycznymi modelami uczenia maszynowego. Przychodzi to jednak kosztem wydajności obliczeniowej, co przekłada się na wy-

datki - wymagane jest posiadanie lub wynajęcie wydajnej karty graficznej z pojemną pamięcią VRAM.

4.2 Pozycja względem konkursu

- **Zwycięzca konkursu (Anastasia)**: 0.8223
- **Oficjalny BERT baseline**: 0.667
- **Nasz wynik**: 0.724
- **Odstęp od zwycięzcy**: -0.0983 punktów
- **Szacowana pozycja**: 18. miejsce w konkursie

5 Analiza wyników

5.1 Co działało

1. **Optymalizacja hiperparametrów**: Systematyczny grid search dał wymierne korzyści
2. **XGBoost**: Lepiej radzi sobie z wielowymiarowymi rzadkimi danymi niż LogReg
3. **Ważenie klas**: Kluczowe dla radzenia sobie z niezrównoważonymi klasami
4. **Smart ensemble**: Ważenie głosów modeli na podstawie ich jakości lepsze niż proste głosowanie większościowe
5. **Transformatory**: Gotowe, wstępnie wytrenowane modele, które następnie dostrajaliśmy

5.2 Co nie działało

1. **Inżynieria Cech**: Dodatkowe cechy pogorszyły wyniki
2. **Trigramy**: (1,3)-gramy gorsze niż (1,2)-gramy
3. **Over-engineering**: Złożoność nie zawsze przekłada się na lepsze wyniki

5.3 Analiza błędów

Najczęściej mylone były klasy:

- **Zagrożenia**: allergens fraud i biological organoleptic aspects
- **Produkty**: semantycznie podobne kategorie
- **Problematyczne klasy**: migration (F1=0.000), other hazard (F1=0.426)

6 Podsumowanie

Projekt wykazał, że transformatory z rozsądnym dostrojeniem, mogą dawać konkurencyjne wyniki w porównaniu z baseline'ami konkursu. A także pokonują jakość klasycznych modeli uczenia maszynowego. Nasze najlepsze rozwiązanie (dostrojony model Qwen3) osiągnął wynik 0.724, lepszy od podanego baseline'u BERT-a (0.667) przy znacznie niższych kosztach obliczeniowych.

Kluczowe sukcesy obejmują:

- **21.1% poprawę** względem oryginalnego baseline'u
- **Systematyczne podejście** do optymalizacji hiperparametrów

Projekt stanowi solidną podstawę dla dalszych badań w automatyzacji klasyfikacji zagrożeń żywnościowych i pokazuje, że nie zawsze najnowsze techniki deep learning są niezbędne do osiągnięcia dobrych wyników w praktycznych zastosowaniach NLP.

Źródła i dostępność kodu

1. Randl, K. et al. (2025). SemEval-2025 Task 9: The Food Hazard Detection Challenge. *arXiv preprint arXiv:2503.19800*
2. Oficjalna dokumentacja konkursu: `https://food-hazard-detection-semeval-2025.github.io/`
3. **Kod projektu**: `https://github.com/cncPomper/NLP-food-hazard-detection`
4. Dokumentacja scikit-learn: `https://scikit-learn.org/`
5. Dokumentacja XGBoost: `https://xgboost.readthedocs.io/`
6. Dokumentacja Qwen: `https://www.kaggle.com/models/qwen-lm/qwen-3`
7. Dokumentacja BERT: `https://huggingface.co/`