Evaluarea complexității cuvintelor

Popa Robert-Daniel, Georgian Sofronea

Abstract

Folosind un set de date care conține un token și un context, încercăm să determinăm complexitatea acelui token (ideal, un cuvânt).

Folosind diferite modele și seturi de caracteristici (features), am concluzionat mai multe lucruri, printre care faptul că augmentarea datelor este necesară. De asemenea, nu am reușit să extragem un procent semnificativ de informație din frază (context), ci mai mult din token.

Modelul antrenat este un ansamblu de tehnici de Gradient Boosting (CatBoost + LightGBM), unde unele dintre cele mai importante input-uri sunt frecvența cuvintelor într-un corp mare de text și variate caracteristici într-un spațiu ce codifică cuvintele (embedding space).

În continuare prezentăm soluția găsită, precum și pașii intermediari prin care am ajuns la aceasta.

1 Caracteristici (features)

1.1 Caracteristici imediate

În primă instanță, am început printr-un proces de analiză de date: am încercat să punem în grafice fiecare caracteristică pentru a vedea eventualele corelații.

Printre primele caracteristici verificate au fost: lungimea cuvântului, număr de silabe, număr de vocale, raport număr vocale la număr consoane.

1.2 spaCy

Am extras cu **spaCy Romanian News Large**: OOV (is out of model vocabulary?), lemma (forma din dictionar), partea de vorbire, lungimea vectorului din embedding space și 10 vectori obținuți prin Principal Component Analysis.

Principal Component Analysis (PCA) este o tehnică care folosește covarianța și valorile proprii pentru a reduce dimensionalitatea. În cazul nostru, am redus spațiul de la câteva sute de dimensiuni (300) la 10, ceea ce a ajutat în procesul de antrenare.

Am observat că partea de vorbire, dacă este necunoscută sau substantiv propriu, corelează cu un scor mai mare. Importanța finală este minoră, dar totuși contribuie.

OOV, deși avea corelație destul de mare, s-a dovedit a avea importanță nulă în analiza finală.

Am mai încercat să extragem informații despre relații dintre cuvinte folosind arborele sintactic din spaCy, dar nu a avut rezultate pozitive.

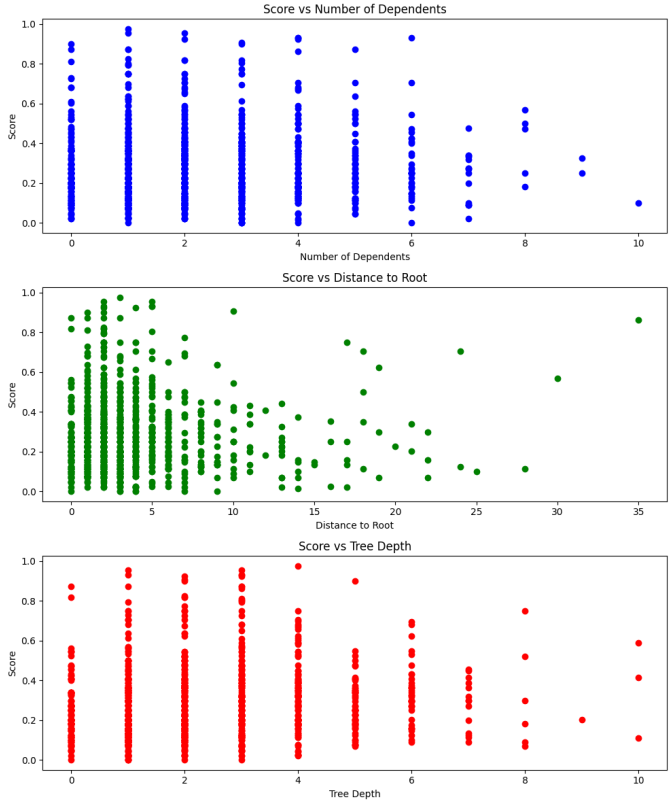

Figure 1: Analiza sintactica

1.3 Corolla

Din corpusul **Corolla** am extras frecvența fiecărei lemme a fiecărui token și am normalizat-o (figura 2). Initial am scos outlierii (acei tokeni ai căror lemme nu au fost găsite în corpusul Corolla; figura 3). Corelația era clar una liniară. Corelația Pearson avea o valoare de 0,56. Am centrat datele si am dat outlierilor valori medii. (figura 4)

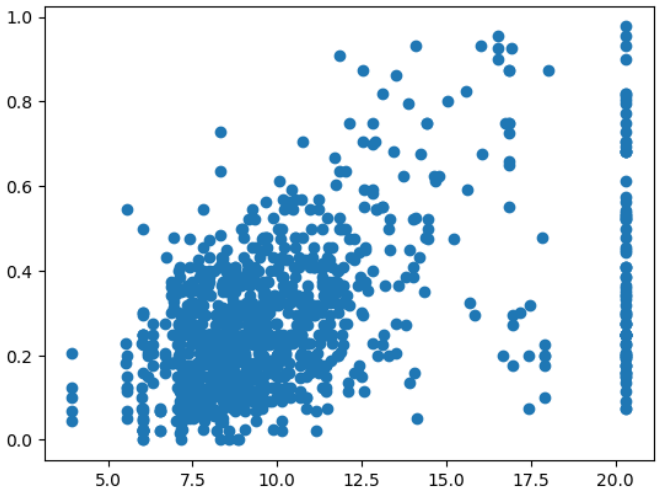

Figure 2: Corelație frecvență în Corolla

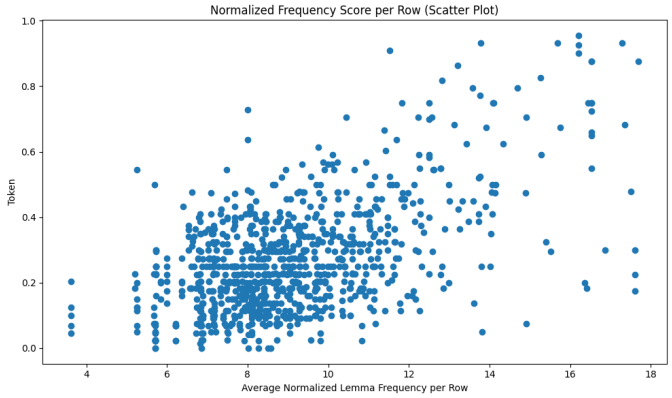

Figure 3: Corelație Corolla fără outlieri

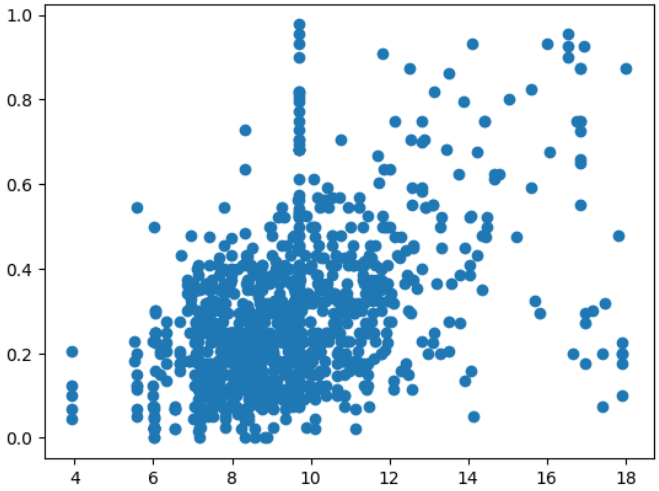

Figure 4: Corelație Corolla cu outlieri mutați în medie

1.4 Embedding-uri extrase

Am extras embedding-uri folosind **word2vec**, **fasttext** și un model preantrenat de tip **BERT**, pe care l-am numit generic - *transformer*. Aceste embedding-uri au fost de asemenea reduse prin PCA la 10 dimensiuni, cu excepția celui din BERT, care a fost redus la 20 de dimensiuni.

1.5 Wikipedia scrape

Folosind `wikiextractor API`, am extras tot corpusul Wikipedia în limba română, căruia i-am aplicat aceeași procedură a corpusului **Corolla**.

1.6 Semantică geometrică

Știind că diferite operații se pot efectua în embedding space-uri, cum ar fi:

$$E(\text{King}) - E(\text{Queen}) = E(\text{Man}) - E(\text{Woman})$$

am dedus că prin scăderea a două cuvinte sinonime, dar diferite în complexitate, am putea extrage o direcție semantică în spațiu a complexității.

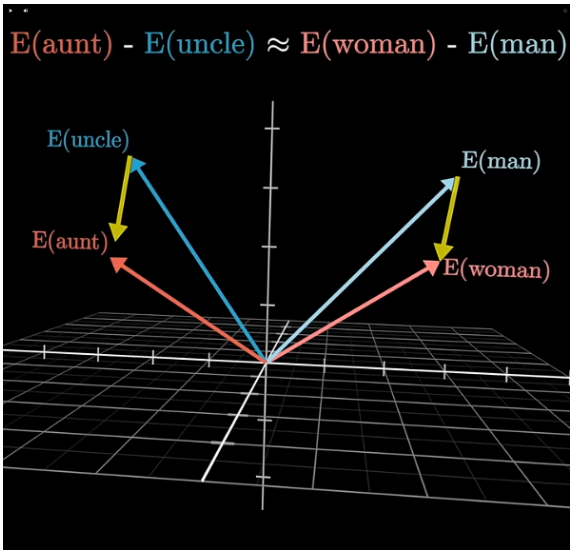

Figure 5: Relații semantice în embedding space

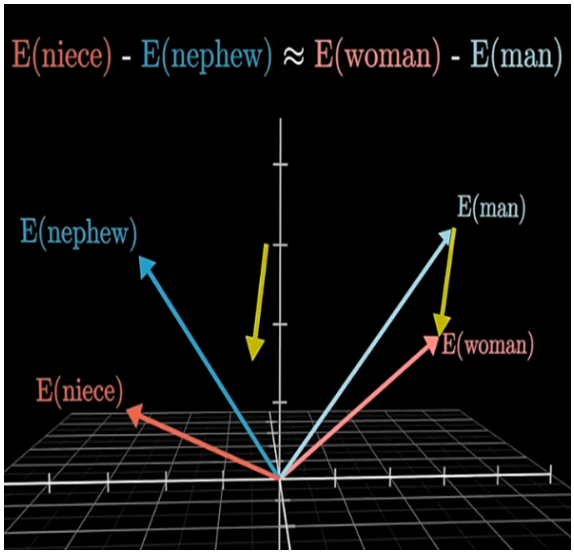

Figure 6: Relații semantice în embedding space

Spre exemplu, $E(\text{achizitiona}) - E(\text{cumpara})$ determină o direcție în spațiu.

Cuvintele fiind sinonime, sunt similare semantic, diferind în mod ideal doar prin complexitate. Am folosit un set de 1000 astfel de perechi de cuvinte (inițial 100) pentru a determina, ipotetic, o direcție a complexității în embedding space-ul BERT.

Mai departe am folosit această direcție (normalizată) pentru a efectua un produs scalar cu embedding-ul token-ului, în speranța că vom obține o metrică a complexității, lucru care a funcționat.

2 Arhitecturi și progrese

Am încercat mai multe modele de machine learning. Am încercat regresia liniară (care inițial a funcționat destul de bine, trecând baseline-ul, doar cu caracteristicile spaCy), după care am încercat un MLP care nu a avut foarte mult succes, având rezultate cu doar 1% mai bune decât regresia liniară.

Am încercat un **KNN Regressor**, în ideea detectării unor relații non-lineare, dar nu a avut succes.

Ulterior, la adăugarea embedding-urilor, am încercat **Random Forest Regressor** și **Support Vector Regressor**, care au avut rezultate similare (35%).

În final am folosit **Gradient Boosting**, dintre care cel mai mare succes a avut **CatBoost (Categorial)**, după care a urmat la mică diferență **LightGBM**, și apoi **XGBoost**, motiv pentru care am ales să le luăm în calcul pe toate, printr-o medie a rezultatului. Astfel am obținut rezultate în jur de 40%.

La descoperirea semanticii geometrice, am observat că **XGBoost** nu se ridică la standardul celorlalte două metode în cazul nostru, motiv pentru care am renunțat la el.

Semantica geometrică avea, ca unică caracteristică printr-un model **SVR**, un scor de aproximativ 17%. La integrarea acesteia în modelul cel mai bun, am obținut un scor de **49%**, care a urmat să crească la cel final de **51.66%** prin mărirea listei de perechi de cuvinte simplu-complex de la 100 la 1000.

3 Analiză importanță featureuri

În figurile 7,8 și 9 am plotat crescător importanța fiecărui feature din modelul final.

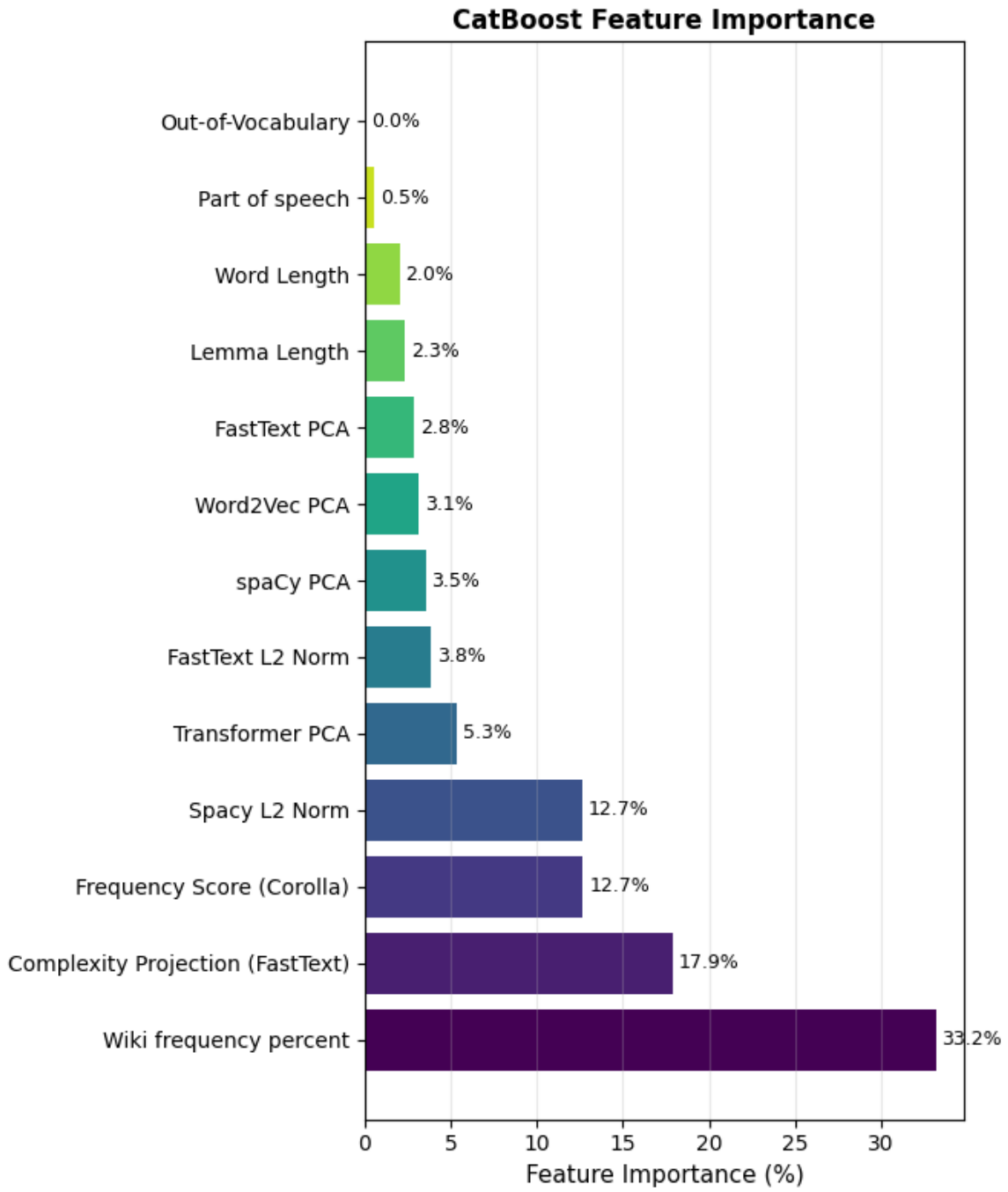

Figure 7: Importanta feature-uri folosite, in CatBoost

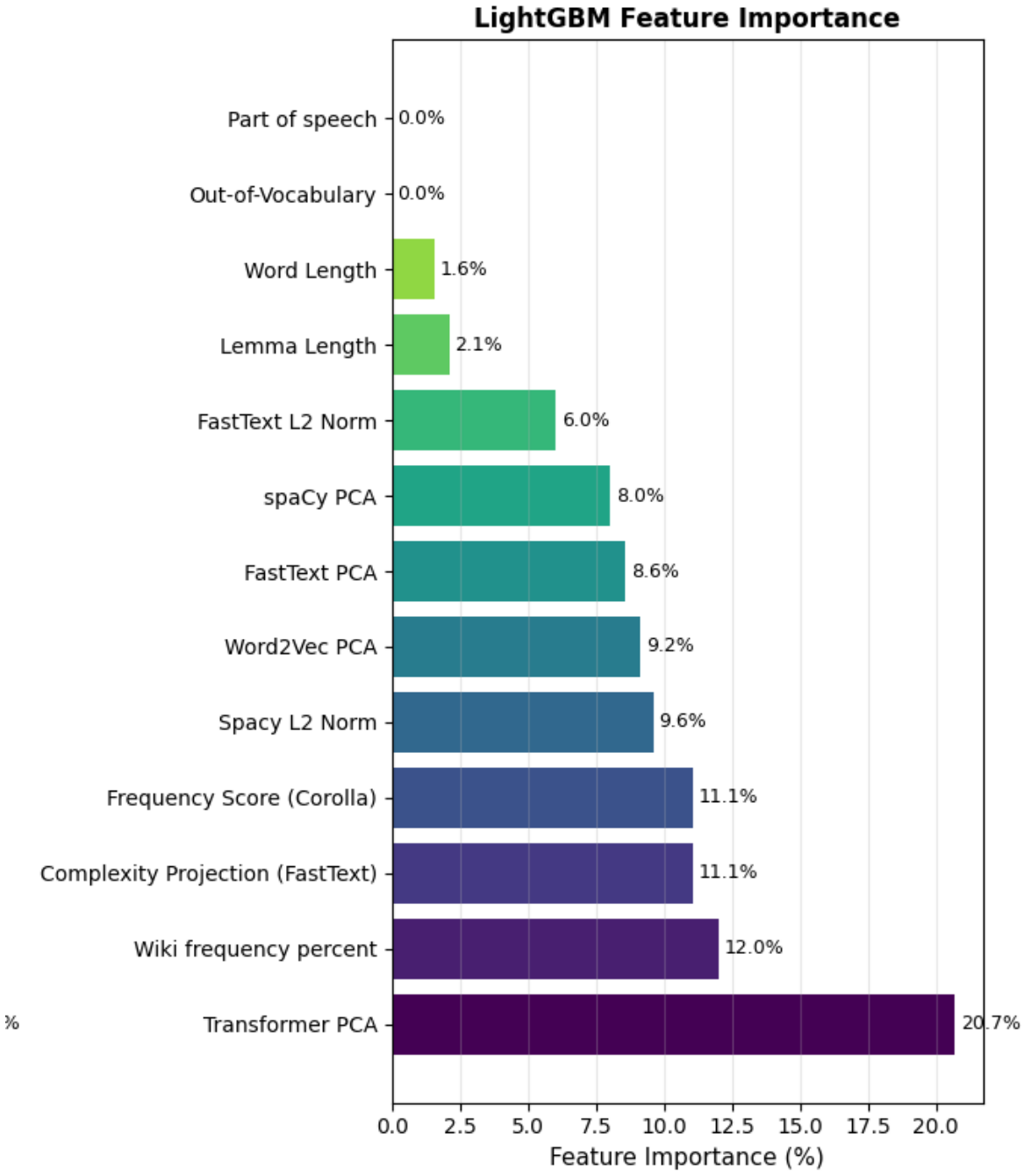

Figure 8: Importanta feature-uri folosite, in LightGBM

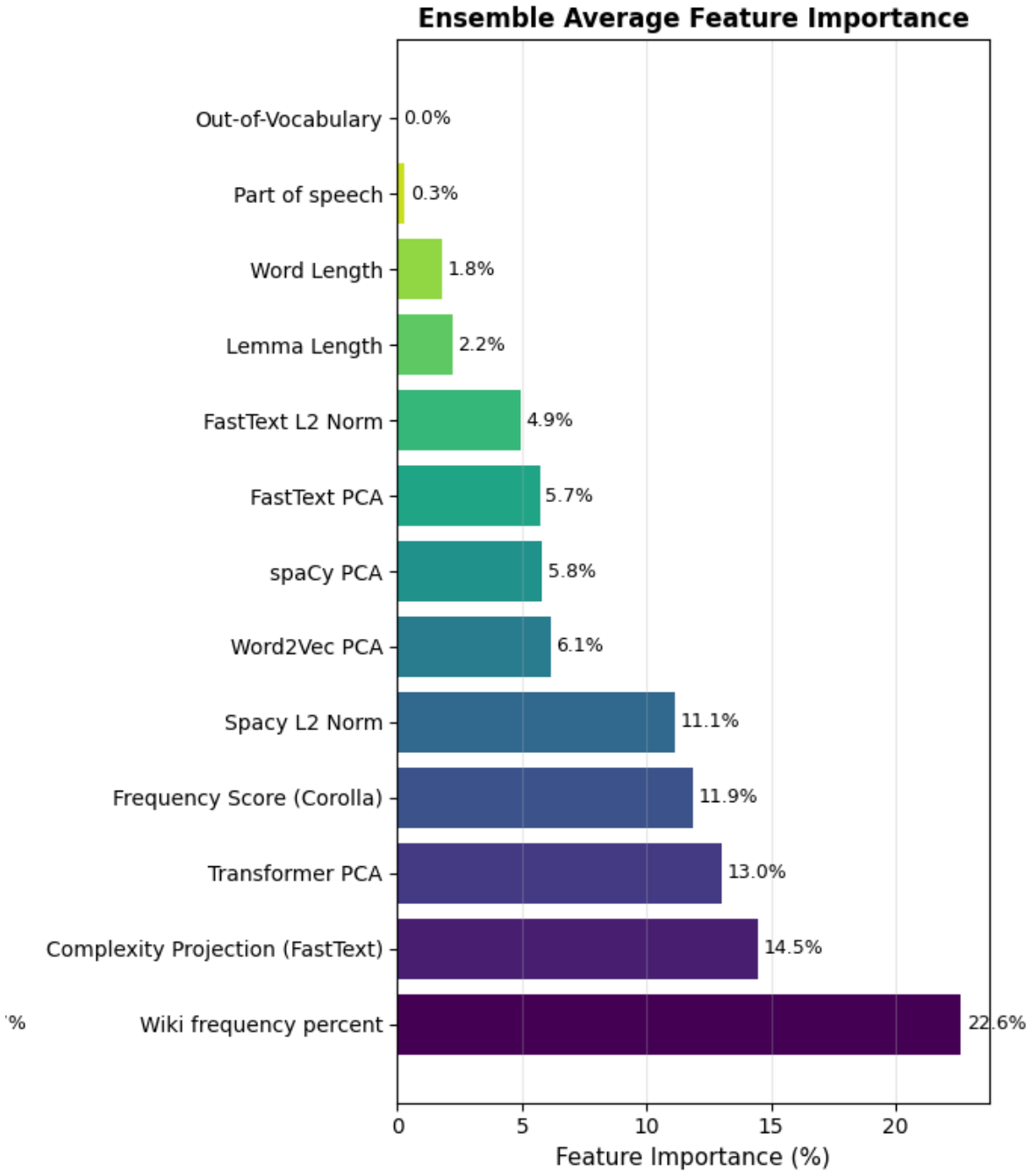

Figure 9: Importanta feature-uri folosite, în ansamblul CatBoost + LightGBM