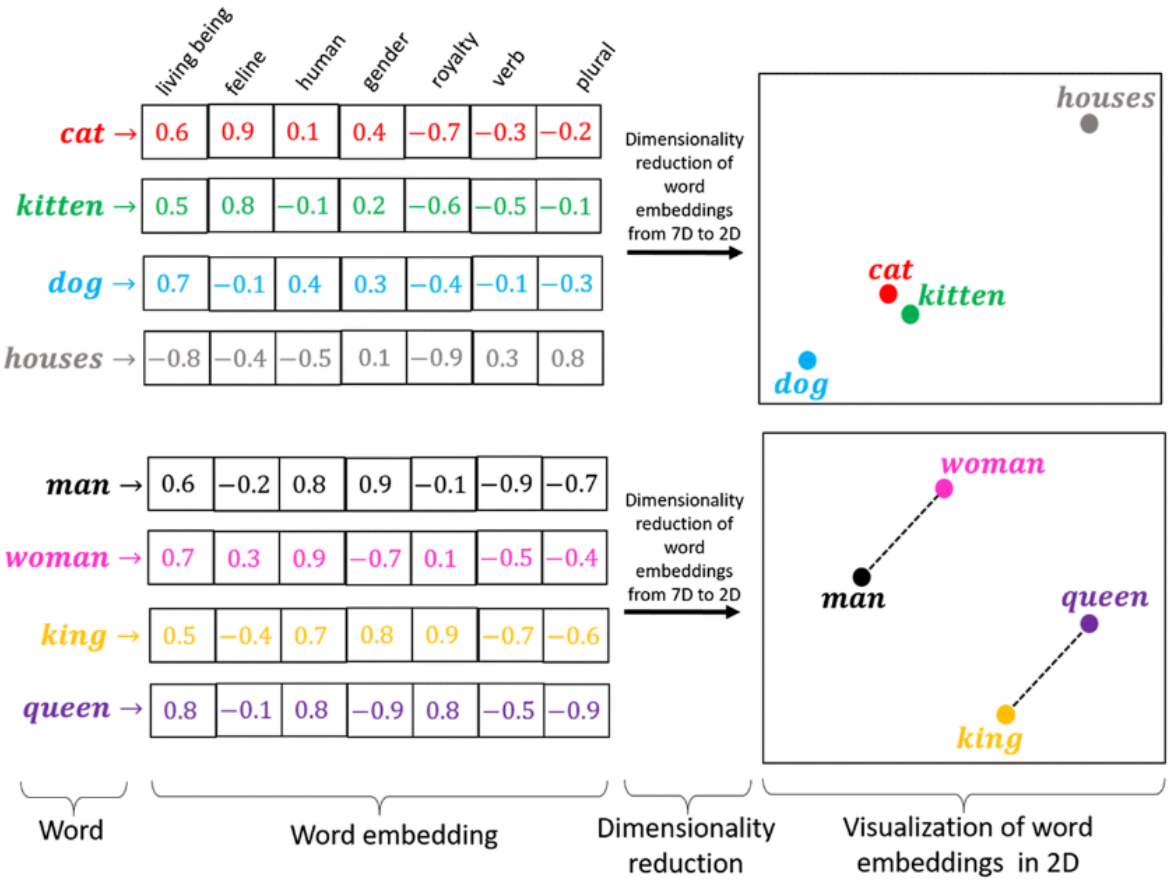
Ce sunt word embeddings?

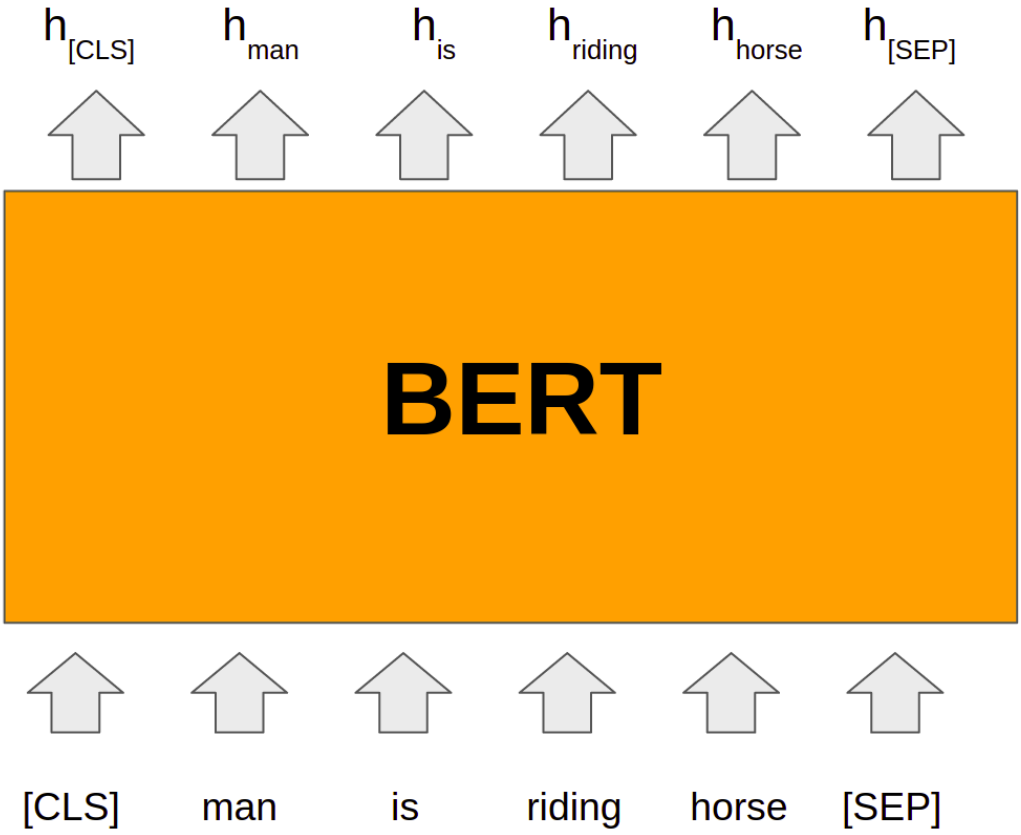
- textele trebuie transformate intr-un format numeric
- cuvintele le vom reprezenta cu niste vectori de features;
- vectorii acestia de features reprezinta ‘semantica’ unui cuvant: cuvinte similare au vectori similari, iar cuvinte nesimilare au vectori departati (vezi cat vs kitten si cat vs dog mai jos)
- acesti vectori sunt invatati de o retea intr-o etapa de preantrenare (mai multe detalii aici https://jalammar.github.io/illustrated-word2vec/)
- daca vrem sa facem clasificare pentru o propozitie, avem nevoie de un vector care sa reprezinte propozitia respectiva
- putem sa facem media vectorilor, insa nu e cea mai buna ‘encodare’, pentru ca nu tine cont de ordinea cuvintelor => putem folosi tot o retea care sa porneasca de la vectori pentru fiecare cuvant si sa invete sa ii ‘agrege’ astfel incat sa tina cont de ordine => BERT



-

Ce este BERT?

- este o retea preantrenata pe foarte mult text, care obtine vectori pentru propozitii, pornind de la vectorii initiali de cuvinte (pentru un overview vezi aici https://jalammar.github.io/illustrated-bert)
- BERT are proprietati foarte bune de transfer learning: porning de la aceasta retea preantrenata, putem sa o finetunam pe un dataset de clasificare care ne intereseaza, obtinand performante mult mai bune decat daca am antrena o retea de la 0
- deocamdata o tratam ca pe un black-box, dar arhitectura pe care se bazeaza se numeste Transformers, mai multe detalii aici https://jalammar.github.io/illustrated-transformer/



- ce inseamna reprezentare contextuala? pentru poza de mai sus vectorul h_{man} agrega informatia de la vectorul ‘man’, dar si de la cuvintele alaturate (is, riding, horse)
- similar h_{riding} este o reprezentare contextuala a lui ‘riding’, care tine cont de man, is si horse
- pentru a face clasificare avem nevoie de o reprezentare a intregii propozitii, peste care adaugam un layer liniar ca la laborator
- o solutie este sa facem media acestor vectori contextuali
- o alta solutie in practica este sa folosim un token special [CLS] pe post de cuvant ‘neutru’;

- BERT agrega vectorul lui [CLS] cu toti ceilalti vectori de cuvinte si obtine o reprezentare a intregii propozitii, care nu e biased catre un anume cuvant (ptr ca [CLS] e ‘neutru’ si nu are o semantica anume).

Resurse de explicat Transformers si BERT:

video short: https://www.youtube.com/watch?v=t45S_MwAcOw
blogs:https://jalammar.github.io/illustrated-transformer/ https://jalammar.github.io/illustrated-bert

Slides

Stanford NLP course https://web.stanford.edu/class/cs224n/index.html#coursework
- Transformers and Pretraining lectures

CMU course
- Transformers lecture https://phontron.com/class/anlp2024/assets/slides/anlp-05-transformers.pdf

BERT presentation
- https://nlp.stanford.edu/seminar/details/jdevlin.pdf

Scop proiect [generic info]:

- antrenat un **baseline** pentru problema noastra: de exemplu un model BERT-like pe datasetul de clasificare ales:
  - pentru acest model se cauta cei mai buni hiperparametri pe setul de validare: learning rate, batch size, weight decay
- imbunatatit **baseline-ul** prin diverse metode:
  - **strategie finetune:**
    - finetune toate ponderile modelului, finetune doar termenii de bias (https://arxiv.org/pdf/2106.10199.pdf) sau finetune doar layer-ul de clasificare
  - **model domain specific:**
    - pornit de la un model BERT-like deja adaptat domeniului target (de exemplu daca facem clasificare legata de Covid, putem porni de la CovidBERT, un BERT preantrenat in continuare pe tweets despre COVID https://huggingface.co/digitalepidemiologylab/covid-twitter-bert-v2)
  - **adaptare model la domeniu target:**
    - daca nu exista un model adaptat domeniului problemei, putem sa preantrenam noi un model BERT pe un dataset specific, folosind obiectivul de antrenare Masked Language Modeling; aceasta retea

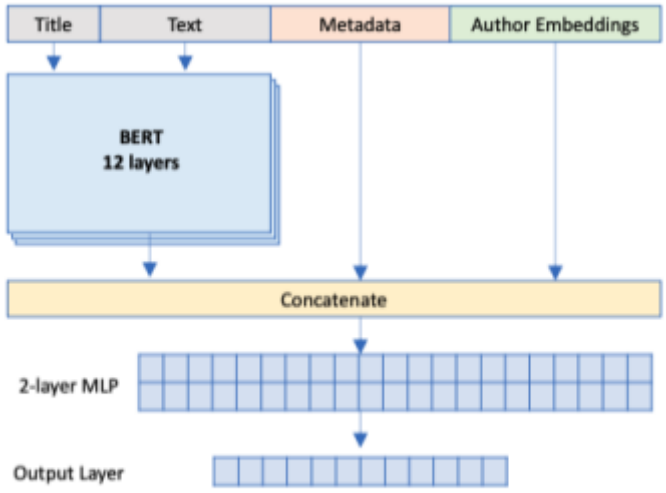
preantrenata o vom finetuna apoi pe datasetul problemei de clasificare care ne intereseaza

- **preprocesare date si augmentare exemple cu alte features:**
  - daca in text sunt prezente emoji-uri, trebuie preprocesate separat, altfel tokenizer-ul nu va recunoaste caracterul

```
import emoji
from transformers import AutoTokenizer
tok = AutoTokenizer.from_pretrained('bert-base-uncased')
x = 'today is a good day 😃'
print(tok.tokenize(x))
demojized_text = emoji.demojize('today is a good day 😃')
print(tok.tokenize(demojized_text))
```

```
['today', 'is', 'a', 'good', 'day', '[UNK]']
['today', 'is', 'a', 'good', 'day', ':', 'grinning', '_', 'face', '_', 'with', '_', 'big', '_', 'eyes', ':']
```

  - in exemplul de mai sus, inainte de a tokeniza textul, emoji-urile au fost convertite intr-o explicatie textuala
  - alternativ se pot extrage emoji-urile din textul original si se pot concatena si tokeniza separat de textul original: astfel se obtine un embedding pentru text si un alt embedding pentru emojis; cele doua pot fi concatenate si folosite ca input intr-un clasificator
  - similar cu de emojis se pot face si alte preprocesari si augmentari de features:
    - extras numere si facut embedding ‘numeric’ separat
    - adaugat alte features discrete (metadate), precum mai jos:



Enriching BERT with Knowledge Graph Embeddings for Document Classification, https://arxiv.org/pdf/1909.08402.pdf

- **ansamblu modele:** combinat predictii de la mai multe modele pentru clasificare:
  - modelele pot proveni de la arhitecturi diferite (de ex. BERT finetunat si un RoBERTa finetunat si facuta media predictiilor)
  - modelele pot proveni de la aceeasi arhitectura:

        - antrenezi 1 model cu dar de la checkpoints diferite, antrenate cu alti hiperparametri cu aceeasi hiperparametri, hiperparametri diferiti sau
- **functie de cost potrivita:**
    - daca problema are clase imbalanced: de antrenat cu o functie de cost potrivita, precum weighted Cross Entropy si incercat diversi parametri de ponderare a clasei minoritare
    - daca este o problema abordata cu retele siameze, se poate folosi un loss contrastiv https://ceur-ws.org/Vol-2936/paper-193.pdf
    - daca datele pot fi zgomotoase, folosit label smoothing https://paperswithcode.com/method/label-smoothing

Poster

- Template poster https://www.overleaf.com/read/hckymqscknpt
- exemplu poster Ad hominem Detection: https://www.overleaf.com/read/dzqwdzxgqvht

Abstract (EEML)

- trebuie scris in engleza, 2 pagini fara referinte
- exemplu https://www.overleaf.com/read/cyvjygkdvphc#b8f50d

Dataseturi (incomplet)

- CT-FAN: A Multilingual dataset for Fake News Detection https://zenodo.org/record/6555293 (cred ca trebuie cont pe Zenodo facut)

Taskuri & dataseturi

Fake News Detection

Competitie: https://sites.google.com/view/clef2021-checkthat/tasks/task-3-fake-news-detection

Dataset (trebuie cerut acces):

https://gitlab.com/checkthat_lab/clef2021-checkthat-lab/-/tree/master/task3 sau https://zenodo.org/records/4714517

daca nu merge obtinut, puteti incerca Covid Fake News detection

Experimente posibile:

- Comparat model BERT antrenat normal vs cu loss reweighting (loss modificat in functie de frecventa claselor https://scikit-learn.org/stable/modules/generated/sklearn.utils.class_weight.compute_class_weight.html#sklearn.utils.class_weight.compute_class_weight)

Model	Results
BERT	
BERT re-weighted	

Covid Fake News Detection

Competitie: https://constraint-shared-task-2021.github.io/

Paper: https://arxiv.org/ftp/arxiv/papers/2011/2011.03327.pdf

Dataset: https://competitions.codalab.org/competitions/26655

Experimente posibile:

- comparat BERT cu CovidBERT https://huggingface.co/digitalepidemiologylab/covid-twitter-bert-v2

Conspiracy detection

Competitie: https://pan.webis.de/clef24/pan24-web/oppositional-thinking-analysis.html

Dataset (trebuie cerut acces in mod normal, dar eu am obtinut deja permisiune deci imi dati un ping sa le shareuiesc; pls dont share them with anyone else, teoretic avem aprobare doar ptr research sa le folosim): https://zenodo.org/records/10680586

- este in engleza si spaniola, faceti splituri de train/valid/test voi

Experimente posibile: Comparat BERT multilingual (https://huggingface.co/google-bert/bert-base-multilingual-cased) cu Spanish BERT (Beto https://github.com/dccuchile/beto) pe datasetul in spaniola

	English dataset	**Spanish dataset**
BETO	aici teoretic n-ar trebui sa se comporte bine, ptr ca a fost antrenat doar pe spaniola	TODO
Multilingual BERT	TODO	TODO

Detect Human or Machine Text

Sunt mai multe variante ale acestui task.

~~**Varianta 1: Dandu-se doua texte A si B, unul scris de om, celalalt de un model, trebuie detectat textul scris de om.**~~

~~Competitie:~~ ~~https://pan.webis.de/clef24/pan24-web/generated-content-analysis.html~~
~~Dataset (trebuie facuta cerere):~~ ~~https://zenodo.org/records/10718757~~

- ~~Daca e suficient de mare bootstrap dataset propus de ei pentru training, putem sa il folosim pe el si sa avem splituri de train/valid/test si sa ignoram pentru moment test setul oficial (care nu e public)~~

~~Model:~~

- ~~se poate folosi un model BERT-like in care sa dai ca input [A, B] concatenate si sa folosesti h_{CLS} ca sa prezici clasa corecta.~~
- ~~se poate folosi o arhitectura siameza, in care encodezi A si B separat cu acelasi model (BERT sau altceva), dupa care concatenezi reprezentarile obtinute $h_{CLS\ A}$ si $h_{CLS\ B}$ si aplici alte layere on top ca sa prezici clasa corecta (sectiunea 3.3.1~~ ~~https://cs231n.stanford.edu/reports/2017/pdfs/801.pdf~~ ~~)~~

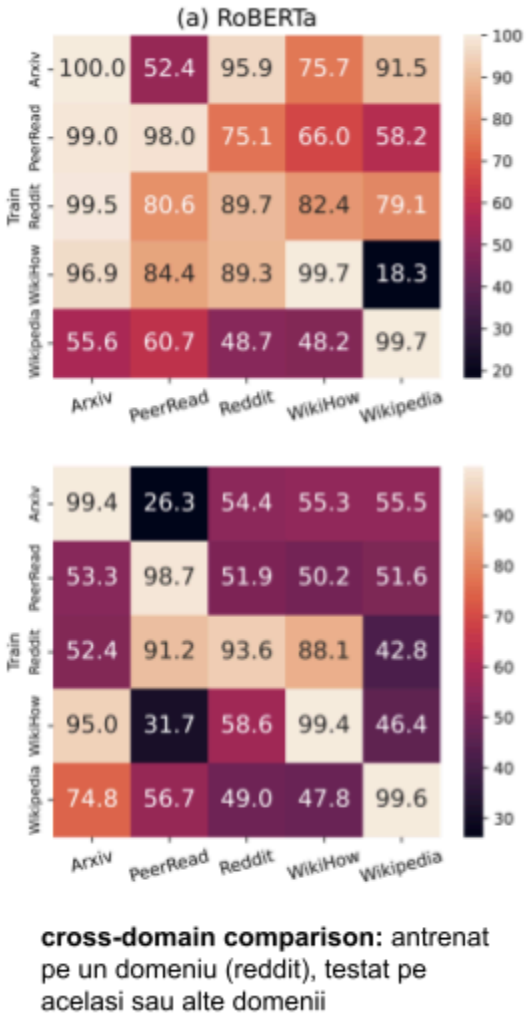
~~Experimente posibile:~~

~~Presupunand ca in bootstrap dataset textele sunt separate si pe domenii, putem testa performanta modelului in-domain (antrenat pe vs cross-domain~~

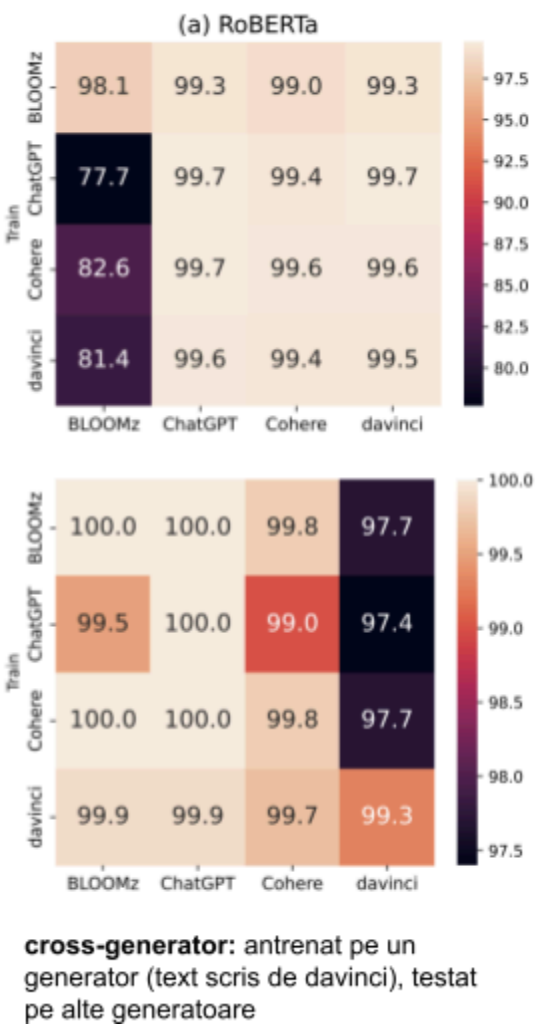
Varianta 2: Dandu-se un text, sa se determine daca este scris de om sau model M

Paper: https://arxiv.org/pdf/2305.14902.pdf
Date: https://github.com/mbzuai-nlp/M4/tree/main/data
Sunt pe mai multe domenii (arxiv, reddit, wikipedia) iar generarile provin de la mai multe modele (bloomZ, chatgpt, davinci, cohere).

Experimente posibile:
Comparat performanta un model M (BERT sau altceva Transformer-based) cross-domain vs cross-generator. Mai jos am decupat din figurile 1 si 2 din paper-ul de mai sus.



cross-domain comparison: antrenat pe un domeniu (reddit), testat pe acelasi sau alte domenii



cross-generator: antrenat pe un generator (text scris de davinci), testat pe alte generatoare

LiRo Romanian NLP tasks

Task: oricare de clasificare din LiRO

Competitie: https://lirobenchmark.github.io/
Modele:

- Romanian BERTS collection https://github.com/dumitrescustefan/Romanian-Transformers?tab=readme-ov-file
- BERT romanian https://huggingface.co/dumitrescustefan/bert-base-romanian-cased-v1 (**antrenat fara diacritice)**
- RoBERT https://huggingface.co/readerbench/RoBERT-base (**antrenat cu diacritice)**

Experimente posibile:

Model	**Liro (dataset original)**	**Liro (dataset cu diacritice**

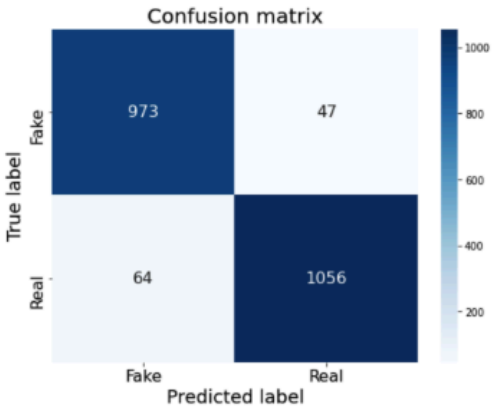
		convertite: ă->a)
BERT romanian (fara diacritice)		
RoBERT (cu diacritice)		

Anomaly detection in network logs

- Anomaly detection -> diferentiere intre instante normale si anormale in setul de date (prin definitie, anomaliile sunt rare)
- Date: https://www.kaggle.com/datasets/solarmainframe/ids-intrusion-csv - trafic de date in care vom considera atacurile ca fiind anomalii
- Primii pasi: XGBoost classifier pe date in supervised fashion
- Masked language modeling pe date (e.g. BERT):
    - Pentru o secventa de antrenare (ce contine features ale unei singure inregistrari), mascam aleator un procent f din features si antrenam modelul sa prezica features mascate
    - Pentru o secventa de test, calculam un scor de anomalie prin mascarea unui procent f din tokenii secventei si agregam (avg / max) probabilitatile date de model pentru tokenii mascati. Intuitiv, daca reteaua a fost antrenata pe un corpus ce contine in mare parte date normale, instantele anormale vor genera o perplexitate mai mare datorita unei corelatii diferite intre features
- ! Transformerii folosesc vocabulare discrete, va trebui sa aplicam binning pentru a preprocesa datele
- Pentru evaluarea modelului, calculam ROC-AUC (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html) intre etichetele corecte (de exemplu 0 = trafic normal, 1 = anomalie) si scorurile de anomalie definite
- O colectie extinsa de metode de detectie de anomalii: https://arxiv.org/pdf/2206.09426.pdf

Evaluare si analiza

- tabele cu modele si metricile relevante task-ului (F1, acuratete, ROCAUC etc.) pe test set
- curbe de metrici si loss pe train&validation datasets
- matrice de confuzie pe setul de test
- analizat exemple misclassified:
    - FP-uri si FN-uri cu confidence mare de obicei ofera indicii despre punctele ‘slabe’ ale modelului
    - de exemplu, FP-ul de mai jos are ‘forma’ unui tweet fake



Predictions examples:
Real tweet predicted as real
coronavirus uk records 20 deaths of people with covid 19 and 3991 new daily cases
Real tweet predicted as fake
coronavirus sir keir starmer says second lockdown would be government failure not an act of god
Fake tweet predicted as fake
novel coronavirus affects only elderly not young people
Fake tweet predicted as real
the number of deaths from respiratory illnesses in ireland this year is comparable to previous years and covid 19 is not causing any more deaths than would usually be

HuggingFace

Puteti folosi biblioteca Huggingface https://huggingface.co/ care gazduieste multe modele Transformer based de de NLP & Computer Vision .

Tutorial Huggingface Transformers (inclusiv finetuning)

https://colab.research.google.com/drive/13r94i6Fh4oYf-eJRSi7S_y_cen5NYkBm?usp=sharing

Sunt multe tutoriale pe net, fie notebooks Colab sau bloguri despre cum sa finetunezi modele. Unele pot fi outdated, puteti porni de la oricare, atata timp cat intelegeti ce se intampla in cod.

Alte exemple de finetunare BERT pe un dataset:
v1:
https://colab.research.google.com/drive/1pTuQhug6Dhl9XalKB0zUGf4FIdYFlpcX#scrollTo=DEfSbAA4QHas
v2:
https://colab.research.google.com/github/abhimishra91/transformers-tutorials/blob/master/transformers_multi_label_classification.ipynb