SATIRE DETECTION BY LMMWORKERS

Team members

Cristea Petru-Theodor

Cringanu Denis Florin

Mihai Alexandru Hutan

Mihai Dilirici

1. ANALIZA DATELOR

- Un prim pas in abordarea noastra a fost sa analizam atent datele si de a incerca sa formam o lista de feature-uri importante de care sa ne folosim in procesul de invatare a unui model.
- O alta idee a fost aceea de a implementa un feature-extraction, insa am observat ca o solutie cu un scor foarte bun a fost realizata in prima ora, lucru care ne-a indicat ca task-ul are rezultate foarte bune pe un model puternic, cel mai probabil un transformer de tip Bert.
- Astfel am ales intr-un final sa nu ne bazam pe feature extraction deoarece puterea modelului Bert: **dumitrescustefan/bert-base-romanian-cased-v1** am considerat-o a fi suficienta pentru acest task.

2A. DATA

- Am hotarat sa antrenam modelul BERT mentionat mai sus pe date urmarind urmatoarele observatii legate de datele disponibile:
  - Concatenarea datelor dupa urmatoarea formula: "[TITLE]" + title.value + " [SEP] " + "[CONTENT]" + content.value
    - Prin intermediul acestei tehnici reuseam cumva sa instruim modelul Bert pentru a invata diferentele dintre date, dintre titlu si content. Am fi putut optimiza enorm procesul in cazul in care am fi reusit sa introducem si o pondere acestor date, spunandu-I astfel lui Bert ca titlul este mai important.
  - Eliminarea unor tokeni care nu ar facilita procesul de invatare (nume de persoane, de exemplu)
  - In cazul in care valoarea content-ului sau titlului era null, luam in considerare doar valoarea disponibila
  - Utilizarea intregului set de date

2B. DATA

- Mai mult, pentru monitorizarea performantelor unui model am decis sa impartim setul de date de antrenament astfel:
  - 80% date pentru antrenare
  - 20% date pentru validare
  - Utilizarea parametrului stratify (deoarece setul de date nu era balansat, folosind un split nebalansat riscam sa avem rezultate neadevarate in timpul antrenarii pe validare / antrenare)
  - Utilizarea "shuffle"-ului

Totodata am decis ca pentru fiecare sample sa realizam mici ajustari pentru a facilita tokenizarea folosind tokenizerului preantrenat al modelului BERT, precum: max_length = 512, padding = True, truncation = True.

Am incercat mai multe variante pentru parametrul max_length, insa acuratetea scadea o data cu acesta, iar valoare de 512 este cea maxima acceptata de modelul folosit.

2C. DATA

- Initial, am optat pentru a nu elimina caractere speciale deoarece am considerat ca pot schimba radical sensul contextual al unei fraze. Spre exemplu, adjectivul "frumos" folosit cu ghilimele poate introduce o nota de ironie, fapt care ar putea ajuta mult in aceasta clasificare binara.
- O prima submisie pe care am incercat-o a fost generata cu un model care a invatat pe o parte foarte mica din dataset-ul de train (5000 de sample-uri) care nu au fost preprocesate. Acest model a obtinut o acuratete de 0.81, dar aici a intervenit o problema pe care nu am constientizat-o initial, faptul ca unele titluri / content-uri erau nule.

2C. DATA

- Cea mai mare greseala facuta de echipa noastra a fost omiterea unei analize in profunzime a setului de date. Tehnica noastra de a concatena titlul si content-ul s-a dovedit a fi o intuitie corecta, insa modelul nu reusea sa invete contextual importanta titlului.
- O alta intuitie pe care am semnalat-o, insa nu am investigat-o din motive de timp si energie, a fost ca datele de test erau foarte diferite fata de cele de antrenare, deoarece modelul avea rezultate foarte bune pe antrenare si validare.

2C. DATA

- Am observat ca dataset-ul este conform SaRoCo asa ca am incercat sa analizam si sa ne ghidam dupa acest paper https://arxiv.org/abs/2105.06456.
- Astfel, prima incercare a fost aceea de a elimina substantive proprii care nu ar reprezente o mare relevanta pentru a permite modelului invatarea unor pattern-uri cu impact real.
- In prima faza, pentru eliminarea numelor, locatiilor, organizatiilor am incercat sa utilizam spacy, dar aceasta tehnica nu s-a dovedit a fi foarte eficienta deoarece era o rata destul de mare a cuvintelor inlocuite ineficient (exemplu de cuvinte incadrate gresit: acolo (person), inainte (location), etc)
- Intr-un final, am incercat sa optimizam aceasta varianta astfel: selectam cuvintele marcate corespunzator de spacy si adaugam conditia da a incepe cu litere mare, fapt care a diminuat foarte mult rata de eliminare gresita a cuvintelor.
- Utilizand aceste tehnici am reusit sa obtinem un scor echivalent cu 0.86.

3. MODELE FOLOSITE

- Initial, pe post de POC, am folosit RoBert pentru a extrage embedding-urile, ca mai apoi sa le folosim intr-un RandomForestClassifier. In pofida optimizarii hyperparametrilor nu am reusit sa depasim un scor de 0.83, asa ca am decis sa folosim embbeding-urile in alt mod.
- In final, am folosit un pipeline deja existent in sdk-ul de la huggingface, mai exact AutoModelForSequenceClassification, caruia l-am dat ca parametru numarul claselor noastre. Acesta adauga dupa embbedinguri doua layere, unul fully-connected cu 768 de neuroni, si functia de activare tanh, iar layer-ul de output are 2 neuroni pe care am aplicat softmax.
- Ne-am concetrat, de asemenea, pe optimizarea hyperparametrilor de antrenare pentru acest model si am obtinut rezultate mai bune decat utilizand Random Forrest Classifier.

4. PARAMETRI DE ANTRENARE

- Am stabilit ca pentru acest task sa mai antrenam modelul deja disponibil si pe setul nostru de date pentru o mai buna acuratete pe setul de testare.

- Toti hiperparametrii pentru fine-tuning au fost incercati in diferite combinatii, noi crezand ca aici se afla de fapt si secretul rezultatelor foarte bune.

- Dintre parametrii custom de antrenare putem enumera:
  - impartirea datelor in batch-uri de 32
  - ajustarea learning-rate-ului la 5e-7
  - folosirea optimizer-ul adamw specific framework-ului pytorch

5A. EVALUAREA/TESTAREA MODELULUI

- Procesul de antrenare parea a fi unul eficient. Am observat ca de la 0.985, intr-un proces de antrenare de 10 epoci, am ajuns la 0.993 pe balanced accuracy pentru datele de antrenament.
- Graficele de loss cat si de balanced accuracy aratau intr-un mod ideal, atat pe antrenare cat si pe validare.
- Antrenarea modelului a fost facut pe un numar de 15 epoci pe toate datele disponibile, iar la fiecare epoca salvam un checkpoint pentru a putea genera o submisie in cazul in care rezultatele de pe validare pareau a fi optime.
- Pentru generarea submisiilor am pastrat aceeasi preprocesare a datelor ca pentru cea a datelor de antrenament si utilizam modelele checkpoint stocate de fiecare epoca.

5B. EVALUAREA/TESTAREA MODELULUI

- In principiu, scorurile obtinute de modelul antrenat se afla in intervalul 0.8400 - 0.8900 utilizand preprocesarea datelor si parametrii optimi pentru antrenare.
- Fara preprocesare, utlizand parametrii default pentru antrenare, modelul reusea sa obtina doar 0.83.
- Utilizand toate datele, modelul preantrenat si tehnica minimalista de a preprocesa datele (concatenarea titlului si a contentului efectuand validarea pentru datele care erau "nan") am reusit sa obtinem o acuratete maxima de 0.8900.

6. BLOCKERS

- Una dintre problemele echipei s-a dovedit a fi increderea in puterea mare de procesare a modelelor de tip Bert si nu ne-am concentrat cat ar fi trebuit pe analiza amanuntita a datelor.

- Am observat intr-un final diferenta mare intre datele de antrenare si cele de test, insa timpul nu ne-a mai permis sa incercam idei noi pe acest lucru.

7. FUTURE WORK

- Pe langa POC-ul prezentat de castigatorii competitiei si de cei din top, acela ca folosirea doar a titlurilor si un max_length mai mic la tokenizare ar fi adus rezultate mult mai bune, venim si cu o alta idee.

- Utilizarea a doua modele, unul pe analiza de titlu și altul pe analiza de content, dupa care aplicare operației | (sau logic) pentru obținerea rezultatului, ar putea aduce rezultate foarte bune, lucru ce nu am mai putut aplica datorită timpului