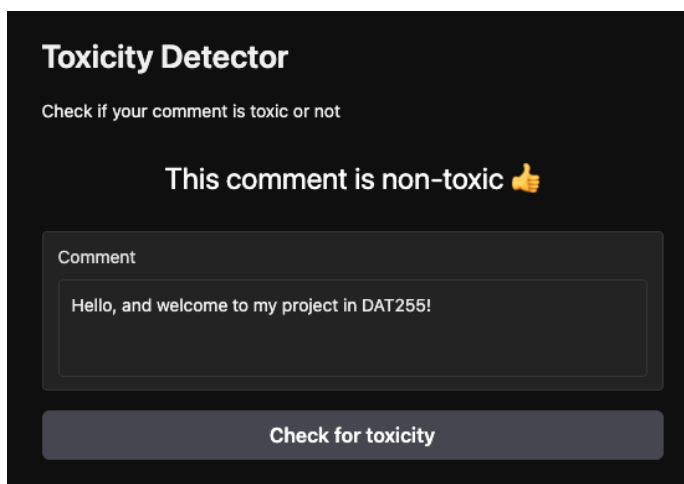


Prosjektrapport i DAT255 – Deep Learning Engineering

Klassifisering av skadelige kommentarer ved hjelp av Deep Learning

25.04.2025

Fredrik Nicolai Crook (kand.nr. 449)



Toxicity Detector

Check if your comment is toxic or not

This comment is non-toxic 👍

Comment

Hello, and welcome to my project in DAT255!

Check for toxicity

Jeg bekrefter at arbeidet er selvstendig utarbeidet, og at referanser/kildehenvisninger til alle kilder som er brukt i arbeidet er oppgitt, jf. Forskrift om studium og eksamen ved Høgskulen på Vestlandet, § 10.

Problembeskrivelse

Målet for prosjektet er å utvikle en dyplæringsmodell for naturlig språkbehandling (NLP) som kan klassifisere nettkommentarer som enten skadelige eller ikke-skadelige (engelsk: toxic vs. non-toxic).

Modellen skal implementeres i et automatisert modereringsverktøy som kan oppdage og merke kommentarer som inneholder fornærmelser, trusler, hatsytringer eller annet skadelig språk. Modellen vil bli evaluert basert på klassifiseringsnøyaktighet (accuracy), presisjon (precision), tilbakekalling (recall), og F1-score, for å sikre en balansert deteksjon av skadelige eller ikke-skadelige kommentarer.

Vi vil starte med tradisjonelle maskinlæringsmodeller som logistisk regressjon og Naive Bayes, og deretter eksperimentere med RNN (Recurrent Neural Network) baserte dyp-læringsmodeller som LSRM og GRU, og transformatorbaserte modeller som BERT for å sammenligne ytelsen.

Det finnes flere eksempler på bruk av dyplæring i tidligere prosjekter som adresserer vårt problem. Dette inkluderer for eksempel *Jigsaw Toxic Comment Classification Challenge* (Kaggle, 2018). Et annet eksempel er Google sitt Perspective API som bruker avanserte dyplæringsmodeller for å klassifisere kommentarer, og er bygget på en transformer-basert arkitektur (Lees mfl. 2022).

Vår oppgaves formål vil ikke være å bringe noe nytt til eller forbedre de eksisterende løsningene, da dette ville krevd enormt mye tid og ressurser. Oppgavens hensikt blir derimot å utvikle en fungerende løsning og forbedre denne gjennom eksperimentering og analyse av ytelse mellom ulike modeller.

All koding vil foregå i Google Colab, da denne tilfredsstiller alle behov for oppgaven samt gir tilgang til nødvendige ressurser som GPU og RAM. All kode vil bli lastet opp i et repository på Github, og lenken til denne finnes i slutten av rapporten under "Andre ressurser".

Data

Vi vil bruke *Google Civil Comments*-datasettet, som er tilgjengelig via Hugging Face (Hugging Face, u.å). Dette datasettet inneholder et stort utvalg brukergenererte kommentarer som er annotert ved grader av skadelig innhold, noe som gjør dem godt egnet for å trene en dyplæringsmodell.

Fordeler og ulemper

Civil Comments-datasettet er omfattende i størrelse og inneholder omtrent 2 millioner kommentarer, noe som gir et solid grunnlag for å trene vår modell. Hver kommentar har flere skade-relaterte labels slik som toxicity, severe_toxicity, obscene, identity_attack, insult, threat, m.m. Selv om vår oppgave går ut på å klassifisere kommentarer med binær klassifisering (enten skadelig eller ikke-skadelig), åpner dette også for mulighet for utvidelse til multilabel-klassifisering. Kommentarene er hentet fra ekte diskusjoner via tidligere Google-prosjekter, som gjør datasettet relevant og representativt for faktiske situasjoner.

Det er også enkelte svakheter i datasettet som er verdt å nevne. Det er langt flere ikke-skadelige kommentarer enn skadelige, noe som gjør datasettet ubalansert. Det er til tross for dette mulig å balansere datasettet ved å fjerne flere av de ikke-skadelige kommentarene, og samtidig ha et representativt utvalg av både skadelige og ikke-skadelige kommentarer grunnet størrelsen på datasettet. En annen potensiell ulempe er at kommentarene i datasettet er noen år gamle (2017 og tidligere). Språkbruk og nettkultur er i konstant utvikling, og modellen vil ikke være i stand til å håndtere nye og moderne uttrykk.

Preprosessering av data

Før trening av modellene ble datasettet preprosessert ved hjelp av standardisering, tokenization og balansering. For basismodellene ble original, tokenisert og paddet tekst sjekket for å verifisere at teksten var blitt tokenisert korrekt.

Alle labels, bortsett fra *text* og *toxicity* var unødvendig og ble fjernet fra datasettet.

```
# Apply clean text
df_civil['text_clean'] = df_civil['text'].astype(str).apply(clean_text)
# Convert toxicity values to boolean (0 if non-toxic, 1 if toxic)
df_civil['toxic'] = (df_civil['toxicity'] >= 0.5).astype(int)
# Drop unnecessary columns
df_civil = df_civil[['text_clean', 'toxic']].dropna()
```

Datasettet ble fordelt i tre deler: *train*, *test*, og *validation*. Treningsdelen utgjør 72%, testdelen utgjør 20% og validering utgjør 8% av datasettet.

Balansering var nødvendig da BERT-modellen ikke ville lære å gjenkjenne skadelige kommentarer etter første implementasjon. Dette førte til at modellen predikerte alle kommentarer som ikke-skadelig. For å løse dette problemet ble det prøvd ut to strategier: *downsampling* og *balansering gjennom klassevekt* (*class weights*). Bruk av *klassevekt* (*Class weight*) løser problemet med skjev klassefordeling ved å fortelle modellen at den skal prioritere den underrepresenterte klassen. Metoden er særlig egnet for store datasett, hvor man ønsker å beholde all informasjon. Ved *downsampling* reduserer man størrelsen på majoritetsklassen for å få like stort utvalg i hver klasse. I dette prosjektet ble det mest hensiktsmessig å benytte *downsampling*, da dette var mulig samtidig som vi hadde nok observasjoner til å trene modellen. Det tillot oss å eksperimentere på mindre datasett om gangen, uten å øke testtiden vesentlig.

På grunn av datasettets omfattende størrelse har vi ikke benyttet eksplisitte teknikker for dataaugmentering. Det ble vurdert at datasettet var variert nok til å trene dyplæringsmodeller uten behov for dette.

Implementering av modeller

Basismodellene som ble brukt består av tradisjonelle maskinlæringsmodeller som Logistisk Regresjon og Naive Bayes og Recurrent Neural Networks (RNNs). Vi eksperimenterte med Recurrent Neural Networks (RNNs), spesielt LSTM og GRU arkitekturer, som er godt egnet for sekvensielle data grunnet deres evne til å opprettholde kontekstuell minne (Géron, 2022, s. 878-883).

Mye av tiden til prosjektet ble brukt til å implementere de ulike modellene. Det ble vektlagt at koden skulle være ryddig og oversiktlig,

Logistisk Regresjon og Naive Bayes

De tradisjonelle maskinlæringsmodellene ble først implementert. Dette ble gjort for å undersøke hvorvidt disse hadde mulighet til å fange opp mønster og i hvor stor grad resultatet ville skille seg fra dyplæringsmodellene.

LSTM og GRU

Den sekvensielle LSTM-modellen er implementert med et innebygd embedding-lag for å lære representasjoner av tekst. Det neste laget er LSTM-lag med 64 enheter og en dropout for å hindre overtilpassing. Modellen er trent i 3 epoker med *binary crossentropy* som loss function og *sigmoid*-aktivering i utgangen, tilpasset binær klassifisering.

GRU-modellen er en enklere arkitektur enn LSTM, men kan til tross for dette fange langsiktige avhengigheter effektivt. Dette gjør at modellen er godt egnet for NLP oppgaver (Srivatsavaya, 2023). Med færre parametere har også modellen raskere treningstid, som gjør den effektiv for større datasett. Modellen er implementert med et embedding-lag, etterfulgt av et GRU-lag. I likhet med LSTM modellen har denne også et Dense-lag med sigmoid-aktivering for binær klassifisering.

Av hensyn til til ble det ikke vektlagt å endre hyperparametere for å optimalisere basismodellene. For videre eksperimentering og tuning vil det være hensiktsmessig å justerende avgjørende parametere - for eksempel reguleringsstyrken (c) for Logistisk Regresjon, smoothing-faktoren (α) i Naive Bayes og dropout- og læringsrate i LSTM- og GRU-modellene.

Transformer: BERT

For å implementere BERT-modellen ble det valgt å benytte BERT Base Model Uncased (`bert-base-uncased`) fra Hugging Face sitt Transformer-bibliotek. BERT egner seg særlig godt for tekst-klassifisering da den er naturlig bidireksjonell. Ifølge dokumentasjonen på Hugging Face sin nettside er modellen “ment for å bli *fine-tuned* på oppgaver der hele setningen (eventuelt med maskerte ord) brukes som grunnlag for å ta beslutninger, slik som sekvensklassifisering, token-klassifisering eller spørsmål-svar-oppgaver” (Huggingface, *BERT Base Model (Uncased)*, u.å, oversatt).

For å muliggjøre raskere trening og eksperimentering, og med begrenset GPU- og RAM-kapasitet, ble modellen først trent på et begrenset utvalg på 20 000 kommentarer over tre epoker. Dette utvalget var allerede preprosessert ved balansering, og ved hjelp av en tokenizer ble den så representert som numeriske sekvenser. Modellen ble konfigurert med Adam-optimalisering og tapsfunksjonen `SparseCategoricalCrossentropy`. Deretter ble modellen trent på et bestemt antall epoker med tilegnet valideringssett.

Evaluering

For evaluering har vi analysert accuracy, precision, recall og F1-score for hver av modellene. Disse metrikkene er brukt for å evaluere hvor godt BERT-modellen presterer opp mot basismodellene. For dette prosjektet har disse metrikkene følgende betydning:

- **Accuracy:** Hvor mange av alle prediksjonene som var riktig.
- **Precision:** Hvor mange av de skadelige prediksjonene som faktisk var skadelige.

- **Recall:** Hvor mange av de faktiske skadelige kommentarene som ble fanget opp.
- **F1-score:** Harmonisk gjennomsnitt av precision og recall.

$$Precision = \frac{TP}{TP + FP} \quad F1\ score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{2 * (Precision * Recall)}{(Precision + Recall)}$$

$$Recall = \frac{TP}{TP + FN}$$

Precision, Recall og F1-score. Illustrasjoner hentet fra Towards Data Science (2020)

Vi vil i hovedsak se på F1-score for å måle ytelsen av modellene. Precision alene kan være høy selv om recall er lav, som betyr at vi fanger få skadelige kommentarer. Recall alene kan være høy selv om precision er lav, for eksempel ved at alt flagges som skadelig. Accuracy alene kan være høy, men til tross for dette kan modellen ikke fange noen skadelige kommentarer (dersom datasettet er ubalansert).

For basismodellene ble resultatet på F1-score:

Logistisk Regresjon: 0,56

Naive Bayes: 0,1

LSTM: 0,63

GRU: 0,61

Resultatet fra Logistisk Regresjon (fig.) viser en høy accuracy på 0,95 som indikerer at modellen klarte å predikere mesteparten av kommentarene korrekt. Den har til tross for dette en lav Recall på 0,44, som indikerer at den predikerte under halvparten av de faktisk skadelige kommentarene som skadelige

```

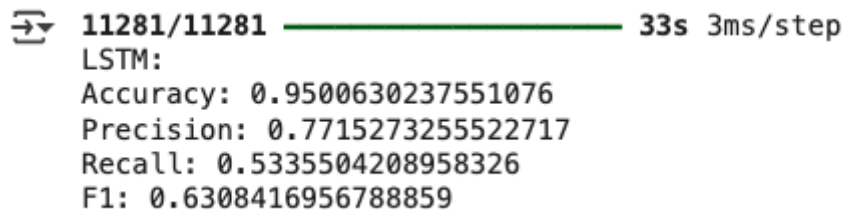
➡ Logistic Regression:
Accuracy: 0.9450682180206386
Precision: 0.776865580198505
Recall: 0.4392558977379014
F1: 0.5611985217641462

```

Med en F1-score på 0,1 yter Naive Bayes modellen dårligst av basismodellene. En recall på 0,06 indikerer at modellen nesten ikke fanger opp noen av de skadelige kommentarene. For å forbedre ytelsen til denne modellen kan det være hensiktsmessig å balansere datasettet.

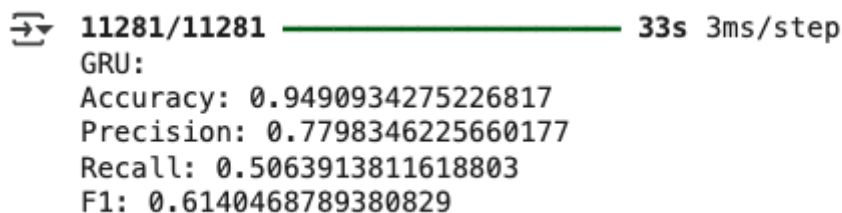
Naive Bayes:
Accuracy: 0.924864602811829
Precision: 0.9560899111343439
Recall: 0.06335954550178405
F1: 0.1188434048083171

Resultatet fra LSTM (fig.) viser en F1-score på 0,63, som er en forbedring fra maskinlæringsmodellene. I tillegg klarer modellen å fange opp flere av de skadelige kommentarene, og har en recall på 0,53.

A screenshot of a terminal window showing the results of an LSTM model. At the top, it says '11281/11281' with a green progress bar and '33s 3ms/step'. Below that, it says 'LSTM:' followed by the metrics: Accuracy: 0.9500630237551076, Precision: 0.7715273255522717, Recall: 0.5335504208958326, and F1: 0.6308416956788859.

11281/11281 33s 3ms/step
LSTM:
Accuracy: 0.9500630237551076
Precision: 0.7715273255522717
Recall: 0.5335504208958326
F1: 0.6308416956788859

GRU-modellen har et tilnærmet likt resultat LSTM, med litt dårligere F1-score.

A screenshot of a terminal window showing the results of a GRU model. At the top, it says '11281/11281' with a green progress bar and '33s 3ms/step'. Below that, it says 'GRU:' followed by the metrics: Accuracy: 0.9490934275226817, Precision: 0.7798346225660177, Recall: 0.5063913811618803, and F1: 0.6140468789380829.

11281/11281 33s 3ms/step
GRU:
Accuracy: 0.9490934275226817
Precision: 0.7798346225660177
Recall: 0.5063913811618803
F1: 0.6140468789380829

For å forbedre basismodellene, kunne det vært hensiktsmessig å balansere datasettet for disse modellene også. Dette ble ikke prioritert, da disse modellene ser på hele datasettet, og ikke bare en subgruppe.

Resultatet for BERT viste en viser en høy F1-score på 0,87 når den ble kjørt på 20 000 kommentarer gjennom 3 epoker. Ved å endre antall kommentarer til 100 000 utgjorde resultatet en svak økning av F1-score til 0,88. Antall epoker ble også justert til 1, da dette viste seg å gi bedre resultat.

Resultatet viser at den presterer bra, både ved å fange skadelige og ikke-skadelige kommentarer.

➔ CLASSIFICATION REPORT	
Overall Metrics	
Accuracy:	0.8804
Weighted F1-score:	0.8804
Toxic Comments (class = 1)	
Precision:	0.8789
Recall:	0.8824
F1-score:	0.8806
Non-toxic Comments (class = 0)	
Precision:	0.8820
Recall:	0.8785
F1-score:	0.8803

Selv om det er oppnådd gode resultater på en begrenset datamengde, kan det være sannsynlig at bedre ytelse ville oppnådd ved bruk av større datamengder, mer finjustering av hyperparametere, justering av Batch-størrelse eller bruk av ensemble-metoder.

For å sjekke at modellen faktisk virket som den skal, ble det gjennomført en subjektiv test der ulike tekster ble klassifisert i selve applikasjonen. Denne evalueringen verifiserte funnene fra de objektive testene, og det ble oppfattet som at modellen klassifiserte korrekt i de fleste tilfeller av setninger med normal oppbygging. Modellen har derimot ikke mulighet til å oppdage mønstre av skadelige kommentarer som den ikke har sett før, og kan ved feil oppdage skadelighet i kommentarer som bruker mønstre som er lignende til tidligere skadelige kommentarer.

Selv om BERT-modellen trener på en mindre del av datasettet, ser vi at den presterer bedre enn basismodellene. I tillegg til modellene som ble testet, kunne vi vurdert arkitekturer som DistilBERT - en mindre og raskere versjon av BERT, men som presterer omtrent like bra. Videre kunne det også vært nyttig å prøve ut RoBERTa for bedre ytelse.

Deployment

BERT-modellen ble implementert i en Gradio-applikasjon. Deretter ble den publisert på Hugging Face sine Free Spaces, som er en permanent og gratis løsning for hosting av applikasjoner. Ved å benytte denne løsningen kan vi enkelt vedlikeholde modellen gjennom Google Colab og deploye den på nytt ved forbedring eller endring av modellen eller brukeropplevelsen.

En alternativ løsning ville vært å implementere modellen som en Google Chrome utvidelse som sjekker kommentarer på utvalgte nettsted som X eller Facebook. Dette ble vurdert, men på grunn av tidsfaktorer og at dette ville gå ut over oppgavens formål ble dette alternativet sløyet og heller noe som kan implementeres ved senere utvikling.

Konklusjon

Vi har eksperimentert med ulike modeller, både klassiske maskinlæringsmodeller og RNNs. Analyse av accuracy, precision, recall og F1-score viser at det beste resultatet ble oppnådd ved bruk av den transformer-baserte modellen BERT.

Ytelsen til basismodellene kunne blitt forbedret ved optimalisering, noe som ikke er blitt vektlagt i dette prosjektet. Resultatet viser imidlertid at dyplæringsmodellene presterte best, med en F1-score på 0,88.

For videre arbeid kan det være nyttig å eksperimentere med flere hyperparametere for å optimalisere modellen, i tillegg til å utforske andre modeller som DistilBERT og RoBERTa eller ensembler. Det kan også være aktuelt å trene modellen på nyere datasett, ettersom språk er noe som er i konstant endring.

Kilder

Géron, A. (2022) *Hands-On Machine Learning with Scikit-Learn, Keras & Tensorflow*. 3. utg. USA: O'Reilly Media. (Digital kopi)

Hugging Face. (u.å) *BERT Base Model (uncased)* Tilgjengelig fra:
<https://huggingface.co/google-bert/bert-base-uncased> (Hentet 24. april 2025)

Hugging Face. (u.å) *Civil Comments*. Tilgjengelig fra:
https://huggingface.co/datasets/google/civil_comments (Hentet 24. april 2025)

Jayswal, V. (2020) *Performance Metrics: Confusion Matrix, Precision, Recall, and F1 Score* Tilgjengelig fra:
<https://towardsdatascience.com/performance-metrics-confusion-matrix-precision-recall-and-f1-score-a8fe076a2262/> (Hentet 24. april 2025)

Kaggle. (2018) *Toxic Comment Classification Challenge*. Tilgjengelig fra:
<https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge/overview> (Hentet 11. april 2025)

Lees, A.W., Tran, V.Q., Tay, Y., Sorensen, J.S., Gupta, J., Metzler, D. & Vasserman, L., 2022. *A new generation of Perspective API: Efficient multilingual character-level transformers*. Tilgjengelig fra: <https://arxiv.org/abs/2202.11176> (Hentet 11. april 2025).

Srivatsavaya, P. (2023) *LSTM vs GRU*. Tilgjengelig fra: <https://medium.com/@prudhviraaju.srivatsavaya/lstm-vs-gru-c1209b8ecb5a> (Hentet 23. april 2025)

Andre ressurser

Lenke til GitHub Repository.

<https://github.com/587855/ToxicCommentsClassification.git>

Hugging Face space:

https://huggingface.co/spaces/fredrikcrook/Toxic_Comment_Classifier