KØBENHAVNS UNIVERSITET
DET NATUR- OG VIDENSKABELIGE FAKULTET

AUTOMATISK FAKTURA KLASSIFICERING MED BERT

BACHELOR I DATALOGI

Skrevet af: Joachim Behrend - tfc551

Vejleder: Anders Søgaard

Hjælpevejleder: Ana Valeria Gonzalez

I sammenarbejde med: Zangenberg & Company

August 16, 2021

Github Repository: https://github.com/JoachimBehrend/Zangenberg

1 Abstract

The modern corporate world is filled with constant transactions, for example, objects and services are bought and exhanged at a high rate; a rate which makes it increasingly difficult for most big companies to keep an accurarate overview of their past spendings. The spendings of a company are often documented in the form of invoices, where each invoice represents an individual transaction. Due to the large amount of incoming invoices, and the lack of training of employees in classifying invoices manually, invoices are often classified into the wrong cost group. This can have considerable negative consequences for a business such as large unforseen or unnessesary expenses, many of which could have been avoided.

As a result the objective of this project will be to create an automatic invoice classification system for a company using a real world dataset and three different machine learning algorithms. The results show that it is possible to build effective classification baselines using standard algorithms such as Naive Bayes and Logistic Regression. In addition, this thesis shows that classification is further improved with more complex transformer architectures. Overall, the work presented in this thesis tackles the real world problem of invoice classification, and describes the implementation of tools and data preprocessing pipelines, which are now being used in production by the project's sponsor, Zangenberg & Company.

Contents

2 Introduktion

I dette projekt vil jeg udforske automatisk fakturaklassifisering implementeret med en af de nyeste og mest banebrydende publicerede algoritmer inden for Natural Language Processing (NLP). NLP er et subfelt inden for datalogien og et vidensmæssigt skæringspunkt mellem kunstig intelligens og lingvistik [22]. Feltet behandler problemer omhandlende computeres genkendelse af menneskelig tale og forståelse af struktur og mening i tekstuelle og sproglige input. Faktura klassificering handler om at identificere, hvilken omkostningsgruppe de enkelte økonomiske transaktioner tilhører. Fakturaklassifikation er basisgrundlaget for et firmas overblik over omkostninger, og gennem dette, optimering og fjernelse af unødvendige udgifter [23]. Projektet vil arbejde med Bidirectional Encoder Representations form Transformers (BERT) [1], som er en Masked Language Model publiceret af Google i 2018. Modellens indflydelse på kvaliteten af et data-drevet faktura klassificeringsystem vil blive undersøgt. Undersøgelsen foretages ved at implementerer klassifikationsystemet med tre forskellige algoritmer - Hovedalgoritmen BERT og de to simplere baselinealgoritmer Naive Bayes og Logistisk Regression [8][9].

I projektets implementationsdel er det første problem som adresseres præprocesseringen af ubehandlet industriel data. Data som, sammenlignet med de datasæt der normalt bruges inden for forskning, er langt fra standardiseret og kræver en stor mængde normaliseringsarbejde før det kan bruges. I projekt opdeles datasæt genereringen i tre skridt: 1) Erstatning af systemnavne og navne på unikke aspekter af firmaet, 2) Generalisering af dataet, 3) Oversættelse af dataet. Det første skridt sikre at datasæt hentet fra forskellige firmaer er sammenlignelige. Det gøres ved at erstatte ord der gentages ofte i et dataset og er meget specifikke for det bestemte firma, med et fast mere generelt udtryk, som herved gengår på tværs af datasæt. Erstatningsprocessen forklares i nærmere detaljer senere i opgaven. Det andet skridt håndeter kontinuitet mellem hver fakturas formatering og fjernelses af støj, som kan forvirre algoritmen. I det tredje skridt oversættes samtlige faktura til engelsk. Det gøres for at korrigerer for store internationale firmaers tendens til at have fakturadata fremstillet på flere forskellige sprog.

Når datasæt genereringsprocessen er færdig, fokuserer projektet på udviklingen af baselines og mere komplekse pipelines for opgaven om fakturaklassifikation. Her bruges de to klassifikationsorienterede maskinlæringsmetoder Logistisk Regression og Naive Bayes, og der eksperimenteres med en mere nylig deep learning arkitektur i form af BERT.

Projektet vil blive afviklet i sammenarbejde med Zangenberg & Company. Et firma som

rådgiver private og offentlige virksomheder i it-strategiske problemstillinger, med focus på it-sourcing, it-organisering, it-økonomi og forhandling. Zangenberg & Company leverer to faktura datasæt fra to forskellige virksomheder til projektet. Derudover leverer Zangenberg & Company hjælp til implementeringsdelen af projektet hvis nødvendigt. Firmaet vil drage fordel ved at få påbegyndt udviklingen af et redskab til brug ved fremtidige konsulentprojekter. Firmaet vil desuden kunne markedsføre på baggrund af både erfaring og aktive projekter med anvendt kunstig intelligens.

3 Baggrund

I dette afsnit redegøres der for projektets teoretiske baggrund. Afsnittet vil beskrive klassifikationsproblemer inden for maskinlæring, elementer fra NLP som er relevante for projektets forsøg og den seneste udviklingen af maskinlæringsalgoritmer med særlig fokus på de tre algoritmer som anvendes i projektet. Derudover vil der kort gennemgås de evalueringsmetrikker som bruges til analyse af algoritmernes præstationer.

3.1 Klassifikationsproblemet

Klassifikation defineres i maskinlæringens verden, som en supervised learning process, hvor et program lærer baseret på data og herefter kommer med forudsigelser af klasser for tilsvarende data. Klassifikation er på den måde en indeling af data i grupper. Klassifikationsmodellen har til opgave at tilnærme en mapping fra input dataet til diskrete output. Målet er at identifere hvilken gruppe hver datapunkt hører til. En klassifikationsmodel tager data som input der består af et vist antal features. En feature er et individuelt målbar element af det fænonmen der bliver observeret [16].

Der er overordnet to former for klassifikatorer. 'Dovne' og 'Ivrige' [10]. En doven klassifikator gemmer træningsdatet indtil testdata gives som input. Klassifiktationen er herefter udført ved brug af det data der relaterer mest til testdatet. Dovne klassifikatorer bruger ikke tid på at træne, men har mere prædiktionstid end en ivrig klassifikator. Eksempler på dovne klassifikatorer er k-nearest neighbors.

En ivrig klassifikator basere sin model på træningsdatet inden testdataet er givet. Den skal derfor hengive sig til en enkelt hyptotese som skal gælde for alle datapunkter i testdatet. Ivrige klassifikatorer bruger meget tid på træningen men kan udføre prædiktioner hurtigt. Eksempler her er Naive Bayes [8], Logistisk Regression [9] og mere nyelige klassi-

fikatorer som transformerbaserede modeller. Disse modeller er brugt i projektet og bliver nærmere beskrevet senere i sektionen.

Klassifikationens input kaldes 'features' og skrives som x. Outputtet fra klassifikationen kaldes 'labels' eller 'targets', og skrives matematisk som y.

3.1.1 Automatisk Fakturaklassifikation

Økonomisk indsigt i et firma er det mest centrale element når det kommer til at skulle optimere sine it-omkostninger [11]. Trods dette har en meget stor del af alle små og store virksomheder ikke et ordenligt klassificeringssystem implementeret til at gruppere faktura når de modtages. Det resulterer i at firmaets forskellige afdelinger alle ender med at have en bunke uordnede faktura, som ikke kan spores tilbage til den oprindelige omkostningsgruppe. Når optimering og it-omkosntingsanalyser så skal laves kræver det en masse tilsidesatte resourser til manuel klassificering. Automatisering kan reducere mængden af manuelt arbejde betydeligt, og spare firmaet en masse arbejdstimer.

Fakturadatasættene er opdelt i features og labels. Hver datapunkt kan have mere end en feature som hver især defineres af en faktura-informationskolonne. Alle faktura i samme datasæt har samme informationskolonner. Kolonnerne kan dog variere på tværs af forskellige datasæt, alt efter hvilken struktur den enkelte virksomhed har valgt for deres faktura. Næsten alle fakturadatasæt har dog en `psp-betegnelse`, `Levarandør` og `Aktivitetsbeskrivelse`, som der derfor genrelt tages udgangspunkt i ved sammenligning af forskellige firmaers fakturadata. En fakturas features representeres derfor som en to dimensionel vektor x og dens tilsvarende omkosntningsgruppe skrives som label y.

3.2 NLP

Faktura classifisering kan blive kategoriseret som tekst klassifikation, hvilket falder ind under feltet NLP (Natural Language Processing). Naturligt sprog er alt kommunikation som tager udgangspunkt i ord - tekst og tale er gode eksempler på naturligt sprog. Alle ord der indgår i tekst eller tale kan er data. Meget af dette dokumenteres og gemmes i databaser. De konstant voksende mængder data er grunden til at NLP har kunne udvikle sig så hurtigt som det har [5].

Problemet med bearbejdelse af naturligt sprog er, at det indeholder mange kombinatoriske muligheder. Meget små ændringer i struktur eller ord kan ændre meningen af

en sætning fuldstændig. Udvikling af vores sprog sker konstant og der forekommer både ændringer og tilføjelser til ordbøger hver dag. Alt det gør naturligt sprog til en tvetydig og til tider uforudsigelig form for data.

3.2.1 Featurization teknikker inden for NLP

Maskinlæring har brug for numeriske features til udføringen af sine beregninger på træningsdatet. For at omdanne tekst til numerisk form udfører man en featurization proces. Det numeriske data kan herefter bruges til at danne en vektor gennem en proces kaldet vectorization. I dette projekt udnytter vi tre featurization processer: TF-IDF [2] og BOW[6] for vores baseline experimenter, og en embedding baserede tilgang i hovedmodellen.

Bag of words laver en vektorer til hver feature, som er på størrelse med antallet af tokens, der indgår i træningdataets ordforåd. Hvor en token kan være et ord, tal eller et specialtegn. En features vektor dannes ved at se på hver token, og inkrementere det index i vektoren som representere denne tokens plads. En vektorrepresentation kan således nemt dannes for hver feature, men man mister information om ordenes placering i forhold til hinanden.

Term Fequency - Inverse Document Frequency (TF-IDF) fungerer ved at beskrive, hvor vigtig hvert ord i et træningssæt er. TF defineres som den frekvens hvorved et ord ses i et dokument. Man kan beregne TF som,

$$TF = (\text{Antal gange ordet indgår i dokumentet}) / (\text{Antal ord i dokumentet})$$

Nogle ord indgår i en stor procentdel af datapunkterne. Disse får en reduceret vægt af TF-IDF ved at der tilføjes en log-operation under udregnelsen af TF. På den måde tager man højde for ord som "det" og "er", som indgår mange gange men ikke giver meget værdi i forhold til klassificeringen. IDF definere hvor vigtig hvert ord er i dokumentet. Jo flere features som indeholder et bestemt ord, jo mindre total værdi vil det ord få i form af IDF. Hvis hver dokument i træningsættet indeholder ordet, vil det ord ende med at have en værdi på nul da det ikke siger noget om det individuelle dokument. I vores projekt ville et dokument svare til de valgte beskrivende kolonner i en enkelt faktura. TF-IDF proceduren bruger derefter Term Frequency værdien og Inverse Document Frequency værdien til at danne en endelig vægt for hvert ord, beregnet som produktet af de to værdier. TF-

IDF danner til sidst en vektor for hver faktura, der viser TF-IDF værdien af alle ord der indgår i den fakturas tekst-data.

Bag of words er en nem metode at bruge, men den tager modsat TF-IDF ikke højde for irrelevante ord. TF-IDF er derfor bedre til at udelade støj fra sine beregninger.

3.2.2 word embeddings

Word Embedding er processen af lade ord med samme mening få lignende vektorrepresentationer. Representationerne gør det muligt at udføre menings-baserede operationer på orderne. For eksempel ville $vec(\text{Paris}) - vec(\text{France}) + vec(\text{Spain}) = \vec{(}\text{Madrid})$. Word Embeddings er især kompatible med Neurale netværk fordi processens output er kompakte vektorer i lave dimensioner. Størstedelen af de neurale netværk er ikke optimale ved udførelsen af udregninger på vektorer af højde dimensioner. Grunden er at niveauet af generaliserings er højere ved lavere dimensioner. Lavere dimentionelle vektorer er altså bedre til tage højde for eventuelle ligheder mellem ordene, hvilket neurale netværk kan udnytte under lærings-processen[7].

Word2Vec er en distribueret representation af ord som numeriske vektorer, hvor den semantiske betydning beholdes. Representationen er baseret på hypotesen om at ord med lignende betyding er præsenteret i lignende kontekst. Matematisk udføres konceptet ved at vælge vektorer for ord med lignende betyding sådan at der også vises matematiske ensartethed ved brug af Kosinus-Lighed funktionen[21]. Modellen er opbygget af få lag neurale netværk, og træner på store datakorpus for at lære at genskabe den lingvistiske kontekst i numerisk form. [12] Der er to overordnede måder at implementerer Word2vec på. Første metode kaldes Continuous-Bag-of-Words (CBOW), hvor netværket forsøger at forudsige hvilket ord der er mest sandsynligt baseret på dens kontekst. Den anden metode kaldes et skip-gram. Konceptet med et skip-gram er det samme som med CBOW bortset fra at netværket i stedet forsøger at finde konteksten baseret på ordet.

I 2014, ét år efter Word2Vec udkom, publicerede Stanford en kompetetiv tilgang til word-embedding. En tilgang de kaldte GloVe [13]. Forskellen ligger primært i metodens håndtering af gentagende forekomster af et ord i samme kontekst. Word2Vec medtager i dette senarie ikke den information som den ekstra gentagelse af ordet bidrager med. GloVe derimod, lægger vægt på at frekvensen af ordene i en bestemt kontekst er en essentiel del af modellens viden.

3.3 Klassifikationsmodeller

Der er en lang list algoritmer man kan bruge til at løse tekst-klassificeringsproblemer. De tideligere forsøg på fakturaklassifisering har brugt klassiske maskinlæringsmodeller. Denne sektion kommer med en kort introduktion til simple såvel som mere avancerede metoder til fakturaklassificering.

3.3.1 Klassiske klassifikationsmodeller

Der fokuseres i denne opgave på Logistic Regression og Naive Bayes da de vil blive brugt som baselines til sammenligning af de nyere neurale netværk baserede algoritmer.

Logistisk Regression Logistisk regression[9] er navngivet efter den centrale logistiske funktion som i modellen bruges til udregning af klassifikationen. Den logistiske funktion er også kaldet sigmoid funktionen. Logistisk regression bruger den logistiske funktion til at tage et reelt tal og omdanne det til en sandsynlighed mellem 0 og 1.

Modellen bruger en ligning til sin representation af data på samme måde som linear regression. Denne ligning tager input x-værdier og kombinere deres værdi med vægtene β til at finde en output værdi y. Et eksempel på en logistisk regressions ligning er følgende,

$$y = e^{\beta_0+\beta_1 \cdot c} \frac{1}{1 + e^{\beta_0+\beta_1 \cdot c}}$$

Her er β_0 en bias og β_1 er koefficienten for den enkelte inputværdi x. Hver feature i vores input data har en tilsvarende koefficient. Det er bias og denne koefficient der læres gennem træningsdataet.

Logistisk regression er oprindeligt ment til binære klassifikationsproblemer. Metoden udregner en sandsynlighed for at en datarække tilhører en bestemt klasse, ved brug af den logistiske regressions ligning. Hvis denne sandsynlighed er 0.5 eller derover vil rækken blive klassificeret som værende den klasse, hvis ikke vil datapunktet i stedet tilhøre den modsatte klasse. Ved brug af Logistisk regression til muliklasse klassificeringsproblemer anvendes en variation af tilgangen. Her indeler man problemet i en række mindre binære problemer hvor man kan bruge den originale logistisk regression tilgang. For hver klasse ser man på sandsynligheden for at datapunktet tilhører denne klasse i forhold til om den tilhører en blandt alle de andre klasser.

β koefficienterne estimeres baseret på træningsdata. Jo mere præcis en estimerering der

udføres, jo tættere på værdien 1, vil den logistiske regressions resultat være, hvis datapunktet tilhøre den efterspurgte klasse. Træningen benytter maximum-likelihood estimation (MLE). MLE er en metode til estimering af parametrene af en sandsynlighedsfordeling ved optimering af en likelihood-funktion. Målet med MLE er at estimere parametrene sådan at det observerede data er mest sandsynligt under den antagede statiske model.

En af fordelene ved brug af logistisk regression til klassifikation er, at den er nem at implementere. Algoritmen er simpel relativt til de nyere neurale netværksalgoritmer, og de beregningsmæssige operationer er derfor også nemmere at forstå og forholde sig til. Algoritmen er meget utilbøjelig til at overfitte lav-dimentionel data. Ulemperne ved logistisk regression er at dens logistiske regressionsligning bliver mindre effektiv ved højere dimensioner og at der her kommer en øget risiko for overfitting. Metoder som regularisering kan dog forbedre algoritmen, hvis antallet af dimensioner i dataet begynder at blive en komplikation.

Naive Bayes Naive Bayes er en famile af algoritmer som er baseret på Bayes formel. De deler den egenskab at hver feature i teksten er uafhængig af hinanden - hver feature bidrager altså med en uafhængig værdi og de forskellige features vægtes lige højt at modellen. Selv om disse antagelser i realiteten ikke altid er sande, fungerer metoden alligevel oftest i praksis. Naive Bayes algoritmer virker ved at udregne posterior sandsynligheden med Bayes formel,

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)}$$

$p(A|B)$ er sandsynligheden for at begivenhed A sker når under antagelse af at begivenhed B allerede har fundet sted. Hvis formlen specificeres til modellen for fakturadata, udskiftes A og B med x og y - hvor x er notationen for den enkelte fakturas features og $y = \{y_1, y_2, y_3, y_4, y_5\}$ er notationen for de fem klasser som fakturaen kategoriseres i. Bayes theorem for faktura poblemet skrives nu,

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$

Den største beregningsmæessige opgave ligger i at finde $p(x|y)$ hvilket også er kaldet 'likelyhood'. Hvis x er en tekststreng, vil $p(x|y)$ beskrive sandsynligheden for at denne tekststreng ses i sammenhæng med en bestemt klasse i træningsdataset. Da sandsynligheden for at en specifik tekststreng forekommer altid er minimal, opdeles 'likelyhood'

ved at kigge på sandsynligheden for hvert ord i tekststrengen. Denne sandsynlighed udregnes med formlen,

$$p(ord|y_i) = \frac{n_{\text{ord}|y_i}}{n_{\text{ord}}}$$

hvor $n_{\text{ord}|y_i}$ er antal gange ordet forekommer i den bestemte klasses fakturaliste og n_{ord} er antal gange ordet forekommer i hele datasættet. Denne udregning laves for hvert ord i en bestemt fakturas fakturatekst og likelyhood findes herefter som produktet af sandsynligheden for hvert ord. Det er Naive Bayes antagelse om uafhængighed blandt features, som gør brugen af produktet mulig. Nu udregnes $p(y)$ kaldet 'prior' som

$$p(y_i) = \frac{n_{y_i}}{n}$$

Hvor n_{y_i} er antal faktura i træningsættet med klassen y_i og n er antal faktura i hele træningsdatasættet. Med disse sandsynligheder ved hånden kan posterior sandsynligheden findes ved at indsætte i bayes theorem. Grunden til at det ikke er nødvendigt beregne $p(x)$ er Naive Baye's 'naive' antagelse om at alle features har samme vægt. Ens vægt betyder at sandsynligheden for at hver feature $p(x)$ vælges er den samme. Sandsynligheden kan udelades fra formlen da den ikke har indflydelse på relationen mellem de endelige værdier af posterior sandsynlighederne. Efter at posterior sandsynlighederne $p(y_i|x)$ for $i = 1$ to 7 er fundet, vil den største af de syv posterior sandsynligheder repræsentere den klasse som er mest tilbøjelig til at være den korrekte, og denne klasse forudsiges af modellen.

Der er et problem med denne metode. Naive Bayes algoritmen danner gennem sandsynlighederne en oversigt over hvor ofte et bestemt ord fremgår i hver klasses fakturaliste. Hvis et ord ikke indgår i en klasses fakturaliste vil det få sandsynligheden

$$p(ord|y_i) = \frac{n_{\text{ord}|y_i}}{n_{\text{ord}}} = \frac{0}{n_{\text{ord}}} = 0,0$$

Det ene ord får herved en meget stor vægt. Hvis en fakturas feature indeholder ti ord, og et af dem har sandsynligheden 0, vil likelyhood sandsynligheden være 0,0. Ligemeget hvor stor en sandsynlighed de andre ord bidrager med vil resultatet være det samme. Løsningen er at give alle ord en ekstra hændelse, så man nu starter optællingen af ordene ved $n_{\text{ord}|y_i} = 1$. Da samtlige ord's observationer øges med én, vil resten af ordenes vægt ikke ændres relativt til hinanden.

Efter at have trænet modellen på træningsdatasættet vil sandsynlighederne for de enkelte ord således kunne bruges til at klassificere test-datasættet.[1]

Naive Bayes Multinomial er en enerkendt supervised algoritme som bruger nogle features til at finde den forventede tilsvarende klasse. Sådan fungere alle supervised algoritmer, og Naive Bayes skiller sig ud i sine to antagelser om uafhængighed og lige vægtning blandt features. Uafhænigheden gør at beregningerne i Naive Bayes er mange gange hurtigere end i andre mere komplicerede klassifikatorer. Naive Bayes fungerer godt med høje dimensioner af input, som for eksempel med tekst klassifikation.

Dens antagelser om uafhængighed er dens styrke men også dens svaghed. Som førnævnt er uafhænigheden blandt algoritmens input sjældent tilfældet. Jo mere komplicerede klassifikationerne bliver, jo sværere er det derfor for Naive Bayes at opretholde en tilstrækkelig præcision.

3.3.2 Neurale netværksmodeller (før 2018)

Et neuralt netværks struktur er baseret på byggeklodser kaldes neuroner. Disse tager nogle input, udfører matematiske operationer på dem og sender et resultat videre i netværket. Operationerne, der udføres i neuronen, er oftest 3-delt. Først sker en multiplikation af input og neuronens tilsvarende vægte, derefter udføres en addition af resultaterne fra multiplikationen og en bias værdi. Til sidst bruges en aktiveringsfunktion på det endelige output. Aktiveringsfunktionen bruges til at transformere outputtet til en afgrænset værdi som er nemmere at arbejde med.

Når flere neuroner sættes i rækkefølge, og på den måde giver én neurons output videre som den næste neurons input, er der tale om 'feedforward'. Hver lag af neuroner, som modtager input i form af de oprindelige input-værdier eller outputs fra tideligere neuroner kaldes for et 'skjult lag'. Når man opstiller et senarie med inputvariable, et antal lag og et resulterende output, har man et neuralt netværk. Netværket defineres i det beskrevne senarie som et feedforward neuralt netværk[17].

Der er publiceret en lang række variationer af neurale netværk. En af disse er Recurrent Neural Networks (RNN) [19]. Denne type netværk bruger sekventiel eller tidsseriebaseret data. En RNN har en hukommelses-orrienteret tilgang til læringsprocessen, hvor

[1] https://medium.com/analytics-vidhya/naive-bayes-classifier-for-text-classification-556fabaf252b

den betinger nye inputs og outputs på netværkets tideligere input- og output-værdier. De klassiske maskinglærings antagelser om uafhængighed blandt features gælder derfor ikke i et RNN. Et andet karaktertræk ved RNN er at de deler parametre på tværs af de forskellige skjulte lag af neuroner. Denne metoder er anderledes fra andre feedforward neurale netværk, som har en individuel vægt dedikeret til hver neuron. RNN's bruger en variation af backpropagation kaldet backpropagation thought time (BPTT), som er specifik til sekventiel data. Princippet med BPTT er det samme som backpropagation med en opdatering af netværkets parametre gennem sammenligning af estimat og det sande label. BPTT afviger dog ved at summere errors for hver tidsserie, hvilket er nødvendigt da vægtene deles af alle neurale lag. Denne summering sker ikke i de andre feedforward neurale netværk som er blevet diskuteret. De mest vellidte typer RNN's inden for NLP har været Long Short Term Memory (LSTM) netværk [4] og Grated Recurrent Units (GRU) [3] som bruger forskellige gate-mekanismer til at beslutte hvilke informationer, der burde huskes over længere sekvenser.

En RNN kan bruge forbindelserne i backwardpassene til at gemme repræsentationer af tideligere inputværdier. Netværket udnytter her det der kaldes netværkets korttidshukommelse. Dette er i kontrast til langtidshukommelse som i NLP ses i form af den gradvise ændring af algoritmens vægte. Mange maskinlæringsaktiviter afhænger meget af netværkets evne til at udnytte korttidshukommelse. Men algoritmerne til identificering af de elementer som skal gemmes i korttidshukommelsen er langsomme og processen kan kulminere i forsvindende gradienter. Gradienterne bruges under backpropogation til at opdatere netværkets vægtes værdier. I visse tilfælde går gradienternes værdierne mod nul indtil man når et punkt, hvor netværkets vægte ikke bliver opdateret længere. Et andet problem kan være det omvendte, at gradienterne går mod uendeligt og at vægtene derfor 'exploderer'. De to senarier kan ske på grund af de beregningsmæssige processer i backward-passet som udnytter 'fintite-precision numbers'. En løsning til forsvindende gradient problemet er Long Short Term Memory (LSTM)[4] neurale netværk. LSTM netværk tilføjer muligheden for at lade gradienter fortsætte gennem backward-passede uændret. LSTM erstatter beregningsenheden i de skjulte neuroner i et neuralt netværk med en LSTM celle. Ud over at have en vægt forbundet til hver neuron er der nu også et LSTM celle stadie. Dette stadie kan af hver ny LSTM celle opdateres, glemmes eller kun en del af det kan vælges at vidreføres. Hvert skridt i backward-passet vurdere nu hvorvidt gradienterne i netværktet fortsat skal ændres, og kan stoppe ændringerne hvis de er ved at resultere i gradienternes forsvinden.

En senere variation af LSTM recurrent neural netværk er kaldet Gated Recurrent Unit (GRU)[3] netværk. Forskellen er at, hvor LSTM har 3 former for gates, har GRU kun en update-gate og reset-gate. Update-gaten bestemmer, hvormeget af de tideligere lærte information som skal viderebringes til fremtidige gates. Reset-gaten bestemmer, hvormeget information der skal glemmes. LSTM celle stadiet er derudover fjernet og inkorporeret i de skjulte vektorer som videregives fra neuron til neuron. På den måde er GRU en forsimpling af et LSTM netværk. Begge former for neuralt netværk er populære løsninger til forsvindende gradienter i en RNN.

3.3.3 Transformer-based models

Hvor RNN's har været det mest dominerende neurale netværk til løsning af NLP opgaver de sidste 10 år, har nyere arbejde introduceret en moderne neural netværksarkitektur, som nu er standarden inden for NLP. Transformer modellen [24] fungerer ved udnyttelse af forskellige metoder og mekanismer. Transformers bruger en sequence-to-sequence (Seq2Seq) arkitektur. Seq2Seq er et neuralt netværk som transformerer én sekvens, til en anden. Dette bruges i oversættelsesopgaver, hvor én sekvens af ord på et bestemt sprog, bliver transformeret til den samme sætning på et andet sprog.

BERT Bidirectional Encoder Representations from Transformers [1] (BERT) er en revolutinerende deep learning model, udviklet af Google i 2018. Modsat andre Language representation modeller, virker BERT ved i præ-træningsprocessen at træne en bidirektionel representation gennem unsupervised learning. Den bidirektionelle tilgang opstår ved at betinge hvert ord's vektorrepresentation på den kontekst der er at finde på begge sider af ordet. Til kontrast fra andre language representation modeller som enten bruger left-to-right kontekst eller kombinationen af left-to-right og right-to-left. Disse to metoder har været meget enerkendte i maskinlæringssamfundet, men mister begge to information ved ikke at tage udgangspunkt i den bidirektionelle retning i forhold til hvert ord. BERT er en empirisk stærk og konceptuel simpel algoritme, som er udviklet af googles research hold. Den opnår rekordtbrydende resultater på de internationale maskinlærinsdatasæt. Efter træning og finjustinging på General Language Understanding Evaluation (GLUE)[20], en kollektion af ressourcer til træning, evaluering og analyser af maskinlæringsalgoritmer, får BERT en præcision på 80,5% hvilket er en forbedring på 7,7% fra den tideligere rekord.

Algoritmen er delt op i to dele - prætræning og finjustering. Prætræningen er en proces ment til at give BERT en forståelse af sprog og sætningsrelationer. Finjusteringen er en

justering af den præ-trænede algoritme til en specifik opgave.

I prætræningsprocessen bruges to unsupervised aktiviteter - Masked LM og Next Sentence Prediction. BERT er udviklet på antagelsen om at bidirektionel sproglæring er mere effektivt end både left-to-right læring og sammenlægningen af left-to-right og right-to-left læring. De betingede læringsmodeller man har arbejdet med indtil videre har kun mulighed for brug af left-to-right og right-to-left læring, da en bidirektionel betingelse ville medføre at hvert ord kunne "se sin egen mening". Det ville være trivielt for modellen at forudse hvert ord korrekt og ingen læring ville være hentet fra processen. I stedet bruges Masked Language Models. En procentdel af ordene bliver tilfældigt udvalgt i hver feature-tekststreng og ordene markeres som `MASKED`. Google har fundet at det optimale valgt er at udvælge 15% procent af ordene til maskering under prætræningen. Algoritmen forsøger at gendanne alle de markerede ords mening ud fra ordets kontekst. Ligesom i en almindelig Language Model gives de markerede ords endeligt udregnede vektorer til en output softmax. Vektorerne udregnes ved brug af en token-embedding, segment-embedding og position-embedding. Til token-embedding har google valgt at bruge WordPiece som har et ordforåd på 30 tusinde ord. Segment-embedding er en binær vektorindkodning af nummeret for den sætning, A eller B, som ordet befinder sig i. Position-embedding er en vektorindkodning af positionen som ordet har i tekststrengen. Med de tre former for embeddings kan man sikre at inputtet til Softmax også indeholder information om ordenes posistion og sætningens oprindelse.

Next Sentence Prediction (NSP) bruges til at give BERT en forståelse af forholdende mellem sætningerne i træningsdataet. NSP vælger to sætninger A og B fra datacorpuset. Disse behøves ikke være to ligustiske korrekte sætninger men er bare en blok af ord. Algoritmen forsøger herefter at forudse hvorvidt sætning B er den næste sætning efter A eller ej. Det rigtige svar er noteret som en binær variabel C med værdier `IsNext` eller `NotNext` ved hvert datapunkt. NSP tager C-variablen som input sammen med de to sætninger. De to unsupervised modeller bruges i sammenhæng med BERT's transformer struktur.

Fordelen ved anvendelsen af denne form for prætræning er det højde niveau af forståelse for sprog og sætninger som man opnår. Ulempen er at tæningen tager længere tid end andre ikke-bidirektionelle algoritmer. Årsagen er at BERT kun bruger 15% af ordenes resultater ved hver iteration grundet Masked-Learning modellen.

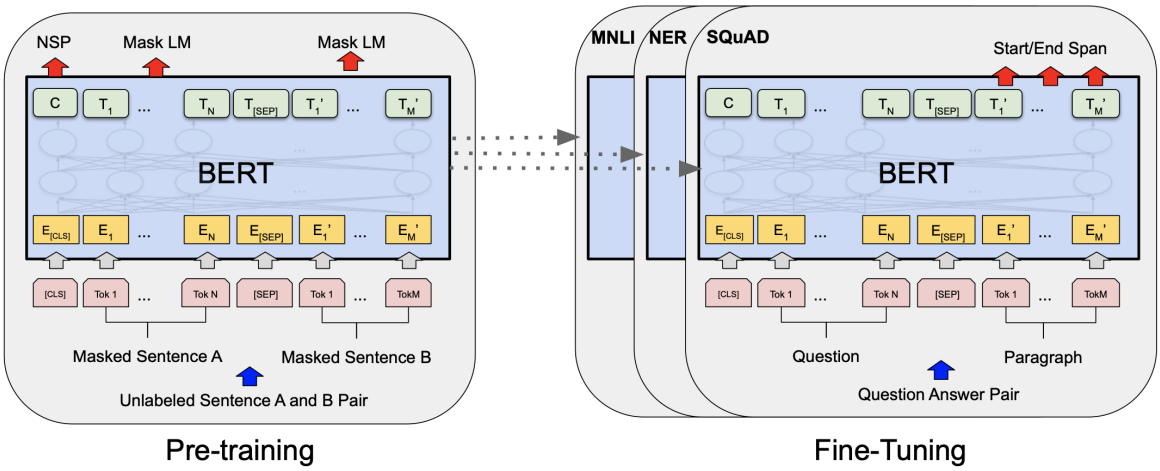

Figure 1: BERT Struktur [1]

Fin-justerings fasen er ment til at kunne eksekveres noget hurtigere da det kun er alle output parametre som skal læres fra bunden af. Alle model parametre undergår en mere beskeden modifisering. Justeingsprocessen foregår med samme aktiviteter som præ-processeringen, men over et mindre datasæt. Her specificeres hvilken form for maskinlærings-aktivitet som man agter at løse. BERT udmærker sig især til sætningsbaserede opgaver som naturligt sprog inferens, sætningsomskrivning og token-baserede opgaver som tekstklassificering, sprøgsmålsbevarelse og enhedsgenkendelse.

Fordelene ved BERT er at man gennem prætræningsprocessen får en mere præcis algoritme, med parametre, der allerede er specificeret til grundig sprogforståelse. Google har allerede trænet de to modeller BERT-base og BERT-large, som man kan hente uden at skulle igennem den beregningsintensive prætræningsproces selv. BERT er helt klart en af de mest kompetetive algoritmer inden for NLP, hvilket også er grunden til at Google har valgt at bruge den som deres seneste søgealgoritme. Den udmærker sig især ved datasæt, hvor dataet indeholder indviklede relationer mellem feautres og klasserne. En ulempe ved at bruge en prætrænet algoritme som BERT er, at der kan være bias i prætræning datasættet som vil påvirke modellens parametre. Derudover er BERT en alogritme af høj kompleksitet, og implementationsprocessen er mere omfattende end de fleste simplere algoritme. Selvom finjusteringen anses som værende en hurtig proces i forhold til prætræningen, vil den stadig tage op imod 4-5 timer på en standard CPU.

3.4 Evalueringsmetrikker

Evalueringen af en models præstation er essentiel for at kende maskinlæringsprojekters fremskridt, succes og finde potentielle svagheder. Evalueringens endelige indblik afhænger dog meget af hvilke evalueringsmetrikker der vælges. De fire metrikker som er vurderet mest relevante for dette projektet er Recall, False Positive Rate (FPR), Precision og F1-score. De har alle det tilfældes, at de tager højde for et ubalanceret datasæt, hvor der ses markant flere forekomster af nogle klasser i forhold til andre.

En populær evalueringsmetrik er accuracy, som representerer forholdet af korrekte forudsigelser i forhold til det totale antal forudsigelser. I bestemte tilfælde giver metrikken dog et overfladisk indblik. Sådanne senarier er projekter med ubalancerede datasæt eller projekter, hvor man har en præference om, hvorvidt størstedelen af fejlklassificeringerne er falske positiv eller falske negativ.

En confusion matrix [14] representerer de fire former for forudsigelser der kan laves, sande positiv, sande negativ, falske positiv og falske negativ. Her vil, hvorvidt forudsigelsen er sand eller falsk, være tilsvarende til om modellen forudsiger den korrekte klasse eller ej. Hvorvidt forudsigelsen er positiv eller negativ afhænger derimod af om modellen accepterer eller afviser nulhypotesen. []

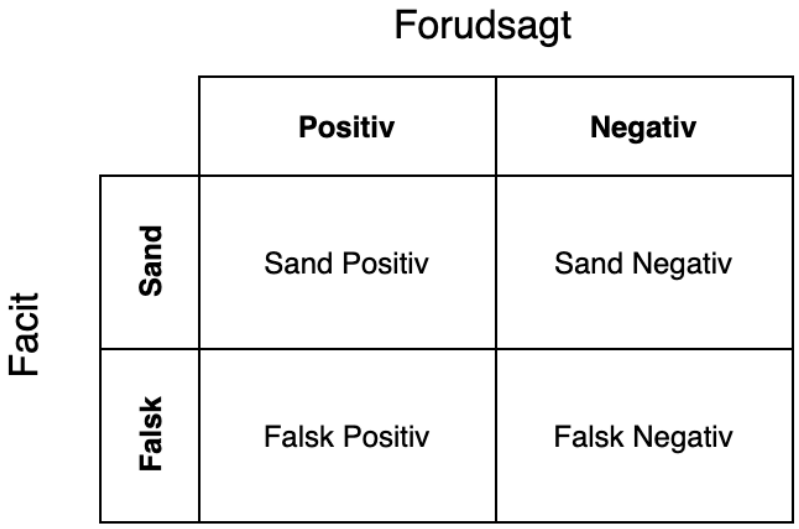

Figure 2: Confusion matrix

Hver klassificering ender i en af de fire grupper. Det endelige antal klassificeringer i

grupperne kan bruges til at udregne Precision, Recall, False Positive Rate og F1-scoren [15].

Precision beskriver forholdet mellem antallet af korrekt forudsagte positive klassificeringer og det totale antal positive forudsigelser. For hver klasse viser precision metrikken, hvor ofte den pågældende model har ret når den forudsiger at et datapunkt tilhører klassen. Precision beregnes som,

$$Precision = \frac{\text{Sande Positiv}}{\text{Sande Positiv} + \text{Falske Positiv}}$$

Recall omtales også som følsomhed, og beskriver forholdet mellem antallet af korrekte forudsagte positive klassificeringer og det totale antal datapunkter som i facit er noteret som positive. Recall fortæller os derfor for hver klasse, hvor mange af de datapunkter som faktisk tilhører klassen som modellen har genkendt og klassificeret korrekt. Recall udregnes som,

$$Recall = \frac{\text{Sande Positiv}}{\text{Sande Positiv} + \text{Falske Negativ}}$$

False Positive Rate (FPR) er sandsynligheden for ukorrekt at afvise nulhypotesen under en klassificering. FPR fortæller derfor noget om, hvor ofte man klassificerer et datapunkt som værende en bestemt klasse uden at det er tilfældet. Af denne grund kaldes FPR også for fall-out eller en false alarm ratio. Med udgangspunkt i confusion matricen, udregnes FPR som,

$$FPR = \frac{\text{Falske Positiv}}{\text{Falske Positiv} + \text{Sande Negativ}}$$

F1-scoren er er et vægtet forhold mellem Precision og Recall. Man kombinerer på den måde de to metrikker til én som tager højde for både falske negativ og falske positiv. F1-scoren er en brugbar evalueringsmetrik i et datasæt, hvor der er tale om mange forskellige klasser og, hvor nogle klassers falske forudsigelser burde have en større indflydelse på resultatet end andre. Dette kunne for eksempel være når datasættet er ubalanceret og underrepresenterede klassers resultater derfor har brug mere indflydelse.

4 Metoder

Faktura klassificering i praksis er to-delt. For det første varierer informationerne i en faktura alt efter hvem der skriver dem. Det vil sige at to faktura fra samme omkostningsgruppe ikke nødvendigvis følger samme indhold eller format. Præprocesseringsprocessen kan bruges til at mitigerer viarationerne og konsekvenserne de har for resultatet. Det andet er, at man ikke er nået til en enstemmighed når det kommer til hvilken faktura model der bruges. Derfor ser fakturaklassificeringer måske meget forskellige ud fra virksomhed til virksomhed, hvilket kan gøre det svært at lave ordenligt konsulentarbejde.

I denne sektion beskriver jeg de dataprocesser der har indgået i projektet, samt uddyber valget af baseline-arkitekturer og hovedalgoritme arkitekturer.

4.1 Data behandling

Ubehandlet faktura data er svært at arbejde med da hver enkelt virksomhed har deres egen faktura formatering. Virksomhedernes fakturaer har en unik struktur og unikke regler for hvad deres tekst indeholder af tegn og ord. Fakturarerne kan af den grund mangle en grad af sammenlignlighed som man kan opveje ved brug af præprocessing. Gennem præprocessering sikrer man nemlig at alle inputs er på samme form. Derudover kan man tilføje funktioner til processen som yderligere kan generalisere de ord, der bruges for visse elementer i virksomheden. Det kan gøres uden at miste præcision eller vigtig viden for algoritmen, fordi det oftest er ligegyldgigt for den, hvilket systemer eller elementer der er snakke om så længe de er af samme type. Firmaer som har problemer med at få overblik over deres faktura er ofte større firmaer som har eksisteret i langt tid og måske endda er nået det internationale markedet. I dette tilfælde vil en stor procentdel af firmaets faktura være oprettet på forskellige originalsprog. Hvis der ikke tages højde for dette vil algorithmen have meget svært ved at sammenligne faktura fra firmaets udlandske afdelinger med de resterende faktura data. Af den grund er det en nødvendighed at implementere et program der kan oversætte de faktura som ikke er på det ønskede sprog - engelsk.

Præprocessing virker således i tre skridt - det første skridt er at erstatte alle instanser af navne i en eksisterende applications liste. Listen er blevet dannet på forhånd af virksomheden med henblik på at skabe en oversigt over navne som indgår gentagende gange i fakturaerne. For hvert navn i listen er der en oversættelse til en bestemt type som navnet passer til. På denne måde kan man få ersattet alle administrative systemnavne med en

generel tekststreng som for eksempel 'administrative systems', alle software programmer med tekststrengen 'software' osv. Efter at genkendt et ord fra applikationslisten erstattes det med en tekststreng indkapslet i møsteret '__tekststreng__'. Ved at bruge et sådan mønster sikre vi at vi nemt kan finde de ovesatte ord senere i præprocesseringen. Dette bringer os til det andet skridt, oversættelse af alle ip adresser, server ip, domaine navne og netværk drev. I tilfældet at en af disse 4 er at finde i en faktura erstattes den af en fast oversættelse, som for ip adresser for eksempel er '__ip_address__'. Erstattede elementer tilføjes til en kolonne i output arket sådan at man mindsker mængden af tabt information.

Det andet skidt i processen er nu at sikre en ensformig formatering. Formateringen manipuleres ved brug af regex, hvor vi fjerner alle uhensigtsmæssige tegn og sikre at alle ord står med småt.

Det tredje skridt er oversættelse af vores features. Først tjekkes om den individuelle faktura er på et ønsket sprog - indtilvidere bruges engelsk. Til at genkende sproget bruges derfor et engelsk biblotek. Hvert ord i en faktura løbes igennem og der undersøges, hvor stor en procentdel af ordene som er på engelsk. Hvis procentdelen er lavere end 60 procent bliver denne faktura markeret som 'incorrect language'. Ellers markeres den som 'sucessfully processed'. Markeringen bruges senere når google-api oversættelsesprogrammet køres på datasættet.

Outputtet fra preprocessing programmet er input excelarket med tilføjelsen af kolonner indeholdende den generaliserede fakturatekst, sprog-markeringen der viser om en faktura er på korrekt sprog eller ej, og eventuelle erstatninger lavet under normaliseringsprocessen. Oversættelsesprocessen af faktura tickets foregår ved at programmet læser det givne output fra præprocessing. Fra denne fil læses alle rækker med faktura markeret som 'inccorect language'. Efter dette bruges api-en først til at identificere hvilket sprog hvert faktura er skrevet på originalt. På den måde ved vi nu hvilket oprindelsessprog som vi skal oversætte fra. Det registrerede oprindelsessprog indskrives desuden i outputtet da det kan bruges som en vigtig ressource i senere benchmarketing analyser. Nu inddeles fakturaerne i grupper baseret på originalsproget således at hver gruppe kan sendes samlet gennem googles api og blive oversat alle sammen på en gang. Det fortrækkes frem for at sende hver enkelt faktura en af gangen da hver komunikationsinitiativ med api-en er meget tidskrævende. Da vores applikations ersattede ord fra preprocesseringen ikke skal oversættes af API'en bruger vi MD5-kryptering til at sikre at ordet ikke findes i en af google-oversæts ordbøger. Efter oversættelsen bruges dekryptering til at få det

indkrypterede ord på normalsprog igen.

4.2 Data indsamling

De fakturadata som er brugt under projektet er samlet af Zangenberg & Company og klassificeret i fælleskab i firmaet. Den manuelle klassificering af hver faktura blev udført over en samlet periode af 10 dage, og udmundede i et datasæt med i alt 19300 faktura, hver klassificeret i én af i alt fem omkostningsgrupper.

20300 faktura er, baseret på erfaring fra tideligere maskinlæringsprojekter, tilstrækkeligt til at skabe acceptable resultater. Et fremmed datasæt fra virksomhedsverden vil indeholde en lang række usikkerheder. Til et faktisk konsulentprojekt, hvor det automatiske fakturasystem skal benyttes på et nyt firmas data, vil der derfor være brug for mere fakturadata til træningen af algoritmen. Det er nødvendigt for bedre at kunne garanterer at der leveres et produkt af høj kvalitet.

Under den manuelle klassificeringsprocess fulgte alle medarbejdere indvolveret en fast klassificeringsprotokol. Hver omkostningsgruppe er blevet nøjagtigt beskrevet, med en detaljeret vejledning til identificering af faktura tilhørende gruppen. Der er afholdt en introduktion i den manuelle klassificering for at sikre at den manuelle klassificering forgik så ensartet som muligt. Mellem de ansatte der klassificerede fakturadata, var der generelt bred enighed, og eventuelle usikkerheder blev taget op i plenum.

4.3 Projektets baselines og hovedalgoritme

Der implementeres i projektet tre automatiske fakturaklassificeringssystem. To af dem bruger simple NLP-algoritmer som repræsenterer vores baselines, og én bruger deep learning hovedalgoritmen. Baselines er brugt for at have et sammenligningspunkt og for at kunne se den praktiske betydning, som den seneste udvikling inden for NLP har på løsninger af tekstklassifikationsproblemer.

Til baselines er valgt de to klassifikationsalgoritmer Logistisk Regression og Naive Bayes. Begge algoritmer er valgt grundet deres simple og roubuste struktur. Algoritmerne er nemme at forholde sig til og nemme at implementerer. Vores baselines kan derfor sammen forventes at give et godt referencepunkt med anstændige resultater.

Projektets hovedalgoritme BERT er valgt på baggrund af dens mange lovende resultater fra tideligere publicerede implementationer af algoritmen på andre NLP opgaver. BERT

fremgår herfra som den mest kvalificerede algoritme, hvis endelige implementering kan give de bedste resultater. Efter en del undersøgelse har det vist sig at der ikke er gjort mange forsøg på implementation af BERT til løsning af automatisk fakturaklassifikation. Projektet vil på den måde kunne bidrage med konkrete resultater til belysning af BERT's præstationer på opgaver som denne. Opgaven vil forhåbenligt kunne hjælpe andre programmører i deres valg af algoritme, hvis de står med et lignedende NLP problem.

Valget af algoritme til NLP projektet skal primært tages på baggrund af kompleksitetsgraden af ens data og de ressourcer, man har til implementation og finjustering. De valgte baselines er at foretrække hvis der er brug for en hurtig tilgang til gode klasseforudsigelser. BERT vil være at foretrække hvis der er interessere i at investere flere ressourcer, og til gengæld kunne bruge den nyeste state-of-the-art maskinlæringsalgoritme. En investering, som primært kræver at det færdige program skal bruges gentagende gange og på relativt store mængder data, for at kunne betale sig.

Ulempen ved at bruge en af baselinealgoritmerne er, at fakturadata-relationerne kunne resultere i at man ender med en stor gruppe fejlklassificerede data-punkter. Datapunkter der, i det tilfælde, ville skulle identificeres og rettes igennem manuel klassifikation. Ved brugen af BERT opnår man forhåbentligt nogle af de mest konkurrencedygtige fakturaanalyser. Konkurrencedygtighed er essentiel i it-konsulentbranchen, hvor man altid er i konkurrence mod de rivaliserende firmaer. En succesful idriftsættelse af Googles nyeste AI-algoritme til Zangenberg & Companys faktura-klassificeringssystem er meget værdifuldt i forhold til markedsføring. En ulempe ved at bruge BERT til projektet er at det kræver en større mængde ressourcer og at resultaterne ville tage lidt længere tid at frembringe under hvert projekt. Firmaet har imidlertid ikke sat en grænse for mængden af ressourcer til projektet. I forhold til at resultaterne ville tage et par timer ekstra at hente, er det intet i forhold til størrelsen af de projekter som klassificeringen vil blive brugt i.

4.4 Klassefordelingen i projektets data

Man ser ofte at en algoritme er meget effektiv når det kommer til at klassificere bestemte klasser, hvorimod den næsten ignorerer andre. Hvordan de individuelle klasser er distribueret i træningsdatasættet har en stor indflydelse. For at sikre at der ikke eksistere underrepresenterede klasser i projektets data, laves der et histogram over fakturadatasettet brugt til træning.

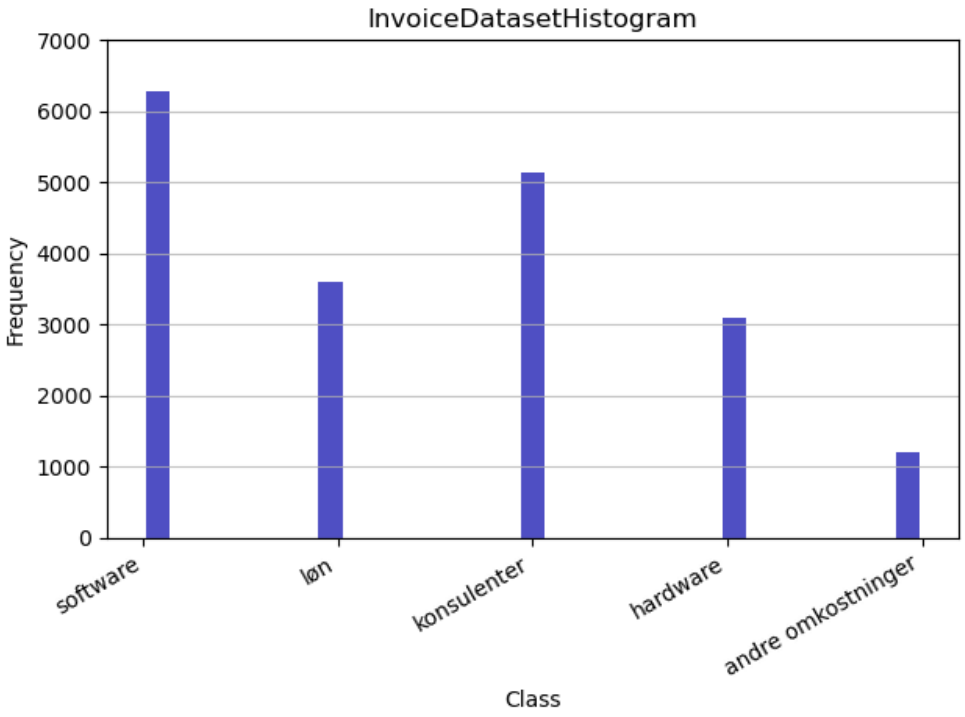

Figure 3: Distribuering af klasser i træningsdata

Fra figuren ses det at der ikke er tale om en normalfordeling. En af klasserne er underrepresenteret (`Andre omkostninger`), og en af klasserne er overrepræsenteret (`Software`). Denne ubalance kan betyde at algoritmerne ender med en mangel på information om de under representerede klasser. Hvis der skal klassificeres et nyt datasæt, hvor en af disse klasser nu i stedet er over-repræsenteret vil resultaterne ikke have samme overordnede præcision. For at få et mere korrekt indblik i algoritmernes ydeevne kunne der anvendes en sampling metode kaldet over-sampling. Her tilføjer man kopier af de underrepresenterede klasser til datasættet. Dette er også kendt som sampling med erstatning. Oversampling er dog ikke

5 Resultater

Forsøget er blevet udført i Python og har taget udgangspunkt i to hoved-biblotekter - `Scikit learn` til baselines og `SimpleTransformers` til brug af deep learning. Scikit learn er et biblotek som både er fleksibelt og brugervenligt, og som understøtter en lang række maskinlærings tilgange. SimpleTransformers er et biblotek som er baseret på `Transformers` biblotektet af Huggingface - et af de førende og mest anerkendte open-source fælleskaber

for forskere inden for kunstig intelligens.

Da dataet er ubalanceret og klasserne ikke er ligeligt distribueret, bruger vi flere forskellige evaluerings måleenheder til at give en mere korrekt forståelse af algoritmernes præstationer. Vi bruger derfor algoritmens F1 score. For hver klasse vises algoritmes Recall , False Positive Rate og Precision.

Den første del af eksperimenter udføres med vores baseline algoritmer Naive Bayes og Logistisk Regression. Her træner vi først algoritmerne på 75% af datasættet og evaluerer den sidste 25%. Under træning af BERT bruges der 4 epoker, en learning rate på 3e-5 og batch size sat til 2.

		f1-score (%)	**accuracy (%)**
Baselines	*Naive Bayes*	77,4	89,2
	Logistisk Regression	78,9	91,0
Deep learning	*BERT*	92,8	96,8

Sammenligningen viser at alle algoritmerne generelt klare sig godt i forhold til accuracy. Naive Bayes og Logistisk Regression får en accuracy på henholdsvis 89,2% og 91,0%. Bert klare sig dog klart bedst med en accuracy på 96,8%. Som tideligere nævnt er accuracy dog ikke en præcis målestok i NLP projekt med ubalanceret data. Hvis der i stedet kigges på algoritmernes f1-scorer, ses det at begge vores baselines præsterer under 80%, hvorimod BERT får en f1-scorer på 92,8%. Når resultatet er betinget på balancen i datasættet og tager højde for underrepresenterede klasser, klare BERT sig noget bedre end de to simplere baselinealgoritmer.

Samme eksperiment blev udført med datamængde brugt til træning som variabel. Målet er at opnå en forståelse for udviklingen i f1-score i takt med forøgelse af mængden af datapunkter.

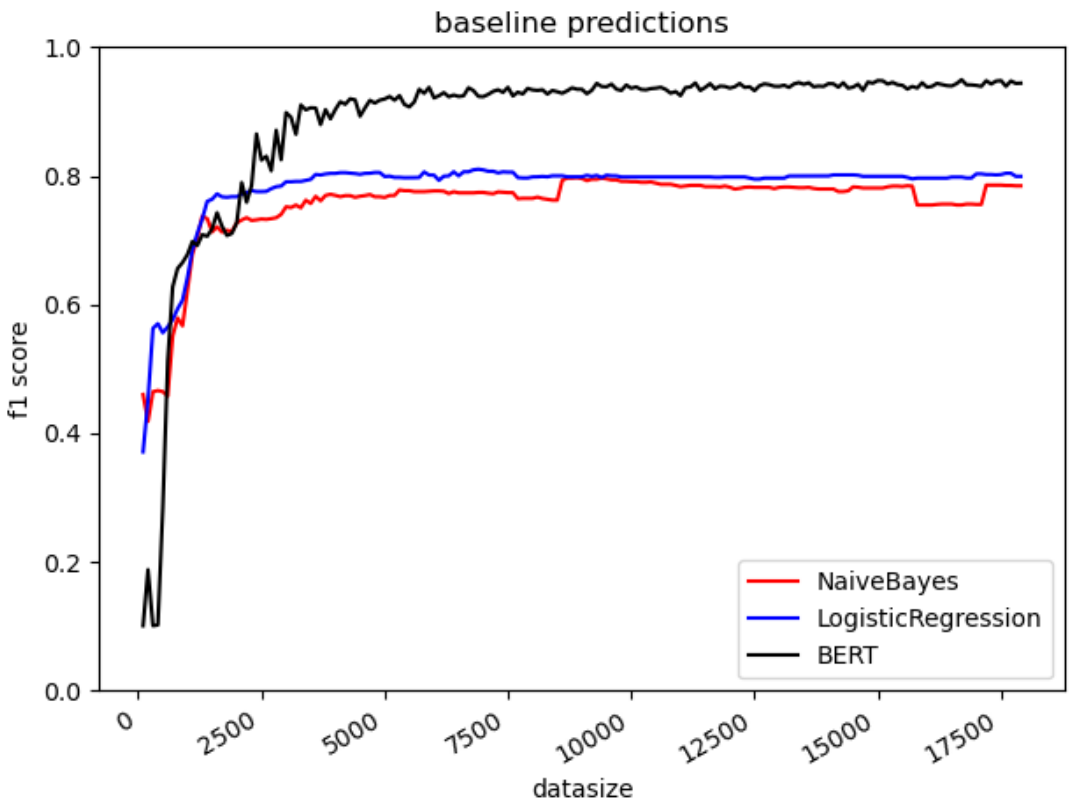

Figure 4: Naive Bayes, Logistisk Regression og BERT - f1 score i forhold til datamængde

Fra graferne ses at begge baselines læringskurver stabiliserer sig meget hurtigt. Dette er især tilfældet for logistisk regression, som allerede ved 2500 når sin maksimale f1-scorer. Herefter ligger logistisk regressionsmodellen stabilt, uden at blive påvirket af de nye tilføjelser til træningsdataet. Naive Bayes læringskurve stabiliserer sig nogenlunde samme sted som Logistisk Regression, med enkelte undtagelser når modellen når 9000 datapunkter og 16000 datapunkter. BERT ender med den højeste f1-scorer men er langsommere til at lære af træningsættet end baselinealgoritmerne. Det ses fra BERT's læringskurve, at modellen først stabiliseres efter 6000-7500 datapunkter. Selv herefter ses let stigning som indikerer at BERT stadig kan finde brugbare informationer i de nye datapunkter.

Algoritmerne er for hver ny fakturamængde trænet på præcis det samme udsnit af datasættet. Mængden og identiteten af evalueringsdataet er konstant i løbet af forsøget, både på tværs af modeller og i takt med forøgelsen af træningdataet.

For at få indsigt i epokernes betydning på BERT-modellens resultater, er der udført et forsøg, hvor mængden af trænings- og evaluerningsdata holdes konstant, men antallet af

epoker nu er en variabel.

Table 1: BERT, epochs betydning på evalueringsmetrikkerne

	1 epoch	**2 epochs**	**3 epochs**	**4 epochs**
f1 score (%)	89,4	91,7	91,9	92,8
accuracy (%)	95,4	96,5	96,6	96,8
cross entropy loss (1e-2)	0,0168	0,0048	0,0042	0,0039

Forsøget viser at BERT får en højere f1-score og accuracy jo flere epoker der bruges. Man risikerer dog at overfitte sin algoritme hvis man træner algoritmen med for mange epochs.

For at få et nærmere indblik i algoritmernes ydeevne med henhold til de enkelte klasser laves der en analyse af Recall, False Positive Rate og Precision for både baselines og Bert.

Table 2: TPR, FPR og Precision for baselines og BERT

	label	**Total samples**	**Recall %**	**FPR %**	**Precision %**
Naive Bayes	software	1720	40,4	0,30	92,4
	løn	1220	99,2	0,02	99,5
	konsulenter	915	97,7	11,5	81,0
	hardware	880	91,3	1,27	95,5
	andre omkostninger	95	46,9,7	0,06	65,2
Logistisk Regression	software	1720	54,5	0,09	85,7
	løn	1220	99,3	0,00	100
	konsulenter	915	97,5	8,50	85,2
	hardware	880	94,3	1,56	94,6
	andre omkostninger	95	34,3	0,39	68,7
BERT	software	1720	95,2	0,19	99,6
	løn	1220	99,8	0.00	100
	konsulenter	915	98,7	2,91	88,8
	hardware	880	95,8	0,32	98,5
	andre omkostninger	95	73,8	0,54	72,9

Tabellen viser at Naive Bayes og Logistisk regression har en lav Recall når det kommer til `software` og `andre omkostninger`. Dette betyder at baselinene har en tendsens til at klassificerer for mange faktura som software. Til gengæld opnås der en høj Precision ved klassificering af klassen `software`. Det fortæller os at selvom modellerne klassificerer for mange faktura som `software`, så finder den også en stor procentdel af de faktura som faktisk tilhører klassen. BERT får relativt også en lav recall i ved klassen `andre omkostninger`. For alle tre modeller gælder det at False Positive Rate stiger ved evalueringen af klassen `konsulenter`. Generelt ses det dog at BERT klare sig bedst ud af de tre algoritmer i forhold til alle tre evalueringsmetrikker.

6 Discussion

6.1 Results

Resultaterne viser at projektets modeller kan træne på præprocesseret fakturadata og opnå gode resultater i forhold til alle de valgte evalueringsmetrikker. Baseline algoritmerne klarer sig godt i forhold til f1-score, hvor Logistisk Regression viser sig at være marginalt bedre end Naive Bayes. Logistisk Regression og Naive Bayes udmærkede sig til hurtig træning og læring af datasættet. Begge algoritmer kunne opnå en høj og stabil f1-score efter træning på få tusinde datapunkter. Den algoritme som scorede best i forsøgene var BERT. BERT klarede sig bedre en vores baselines på f1-score, Recall, Precision og False Positive Rate. BERT udmærker sig ved denne opgave grundet de beskrivende sætninger, hvor ordstillingen og kontekst har indflydelse på hvilken omkostningsgruppe der er tale om. Derudover bruges den prætrænede model `roberta-base` som har en forståelse for generel engelsk sætningsmening og ordforståelse. En forståelse som er opnået ved en blanding af NSP, Mask LM og lag af transformere.

Forsøgende målende f1-scoren i forhold til en gradvis stigning i datamængde, viser at baseline algoritmerne Naive Bayes og Logistisk Regression følger en tilnærmelsesvis ens udvikling. Der er flere årsager hertil. For det første er begge algoritmer implementeret til at bruge vektoriseringsteknikken TFIDF. Vektorrepresentationerne af ordene fra træningsdataet er ens for begge algoritmer, hvilket betyder at de har samme udgangspunkt i forhold til deres beregninger. En anden årsag er at Naive Bayes og Logistisk Regression minder om hinanden ved begge at være linjerer modeller.

Algoritmernes Recall viser at Logistisk Regression og især Naive Bayes har en tendens til at overse for mange `Software`-posteringer og klassifcere dem som noget andet. `Software` er en overrepresenteret gruppe i datasætettet, men Naive Bayes Recall scorer viser at modellen ikke har lært tilstrækeligt om strukturen i faktura fra denne omkostningsgruppe. Da omkostningsgruppen er meget omfangsrig, kan denne mangel på forståelse få store konsekvenser ved brug af Naive Bayes som model til klassificering af ny virksomheds fakturadata.

Fra tabel 2 ses det at samtlige algoritmer opnår en meget høj Precision ved klassifikation af faktura tilhørende klassen `Løn`. Algoritmerne er derved meget effektive til at sikre at en postering faktisk tilhører omkostningsgruppen, og undgå skødesløse forudsigelser. Da alle algoritmer samtidig har et høj Recall ved kategorisering af samme omkostnings-

gruppe, ved vi at algoritmerne også er gode til at genkende faktura tilhørende gruppen. Gennem de to metrikker kan man derfor konkluderer at alle algoritmer har en stor træfsikkerhed hver gang de forudsiger en faktura som værende `Løn`, og at algoritmerne meget sjældent overser en faktura som tilhører `Løn`. Algoritmerne har altså en meget præcis forståelse af omkostningsgruppen, og det er denne forståelse der skal sigtes efter i hver omkostningsgruppe. Denne forståelse kan skyldes at næsten alle faktura som tilhører gruppen, indeholder ordet 'løn' eller 'årsværk', som giver modellerne et godt udgangspunkt for klassifikationen.

6.2 Datas rolle i et virksomhedsbaseret projekt

En af de ting der er blevet tydelige under forsøgsprocessen er, at en meget stor del af selve arbejdet skal ligges i normaliseringen af fakturabeskrivelserne og i at sikre ren data med en minimering af støj. Det data, der er arbejdet på i projektet, er hentet fra virksomheder fra tideligere konsulentprojekter. Mange af de metadata der følger med hver faktura syntes ved eftersyn direkte forkerte, hvilket tyder på et uorganiseret eller lettere overfladisk system til fakturanotering i firmaet. Trods dette, er datasættet af høj kvalitet sammenlignet med andre fakturadata som tideligere er set i Zangenberg & Company. Ny data forekommer ofte uoverskuelig og kompliceret, hvilket er en af grundende til, at udarbejdelsen af en struktureret databehanlingsprocess i et projekt som dette er en værdifuld investering.

6.3 Udsigt for projektet

Projektet har været baseret på en praktiske implementation af maskinlærings- og databehandlingsteori. Produktet og den erhvervede viden opfylder projektets mål og vision, men der findes praktiske udvidelser som burde implementeres før projektet kan anvendes til konsulentarbejde. Udvidelserne involverer inkorporeringen af større mængder klassificeret data, evaluering af modellen på andre firmaers faktura datasæt, og udviklingen af et uddannelsesforløb til effektiv anvendelse af klassificeringssystemet. De 20.000 posteringer der kan trænes på nu giver et indsigtfuldt blik i modellens kvalitet. Men datamængden er ikke nok til at kunne sikre den ønskede kvalitetsikring af modellens forudsigelser. Af samme grund burde en fremtidig tilføjelse til projektet også være nye firmaers fakturadata. Fakturadata som afviger fra den struktur og form set af modellen indtil videre.

Derudover kunne der blive eksperimenteret med optimering af modellens antal af inkluderede features. Disse kan behandles og kombineres på adskillige måder. Undersøgelsen af disse kombinationer ville kunne forbedre modellen og gøre det endelige valg og fravalg af features mere velbegrundet.

7 Konklusion

I dette projekt søgte jeg at finde indflydelsen som den nyudviklede deep learning model BERT har på det praktiske faktura klassificeringsproblem. Ved at bruge teori om NLP-modeller, en struktureret datageneraliseringsprocess og en implementering af hovedalgoritmen samt implementeringen af de to baselinealgoritmer Naive Bayes og Logistisk Regression, undersøgte jeg algoritmernes styrker og svagheder. Baseret på denne metode, var det muligt at måle modellernes konkrete resultater i form af flere velbegrundede evalueringsmetrikker, og sammenligne resultaterne med vores data.

Det var gennem resultaterne muligt at finde konkrete forskelle mellem hovedalgoritmen og vores baselines, og vise hvordan BERT på alle metrikker klare sig bedst. Modellernes implementationer viste at Naive Bayes havde en f1-score på 77,4,%, Logistisk Regression en på 78,9% og at BERT's havde en f1-score på 92,8%. Ved brug af de relevante NLP teorier og projektets resultater over indflydelsen, som deep learning algoritmen BERT har på fakturaklassificering, tror jeg at fremtidig forskning burde kunne undersøge, hvordan BERT modellen håndterer mere diverse og komplicerede fakturadata. Dette projekts resultater ville også kunne udnyttes til videre forskning i udviklingen af det mest effektive automatiske faktura klassifikationsystem i virksomhedslige omgivelser. Projektets fund indikerer, at der stadig er mange nye og effektive metoder at undersøge til udførelse og optimering af virksomhedsrelaterede opgaver.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, 2019.

[2] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. Information processing & management, 24(5):513–523, 1988.

[3] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In Proceedings of the

2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1724–1734, 2014.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

[5] Elizabeth D Liddy. Natural language processing. 2001.

[6] Yin Zhang, Rong Jin, and Zhi-Hua Zhou. Understanding bag-of-words model: a statistical framework. International Journal of Machine Learning and Cybernetics, 1(1-4):43–52, 2010.

[7] Yang Li and Tao Yang. Word embedding for understanding natural language: a survey. In Guide to big data applications, pages 83–104. Springer, 2018.

[8] Daniel Berrar. Bayes' theorem and naive bayes classifier. Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics; Elsevier Science Publisher: Amsterdam, The Netherlands, pages 403–412, 2018.

[9] Maher Maalouf. Logistic regression in data analysis: an overview. International Journal of Data Analysis Techniques and Strategies, 3(3):281–299, 2011.

[10] Adriano Veloso and W Meira Jr. Eager, lazy and hybrid algorithms for multi-criteria associative classification. In Proceedings of the Data Mining Algorithms Workshop, pages 17–25, 2005.

[11] Maryline Gnonlonfoun and Katarzyna Szymczyk. Measures of cost optimization based on the examples of french companies. ZESZYTY NAUKOWE POLITECHNIKI CZĘSTOCHOWSKIEJ RESEARCH REVIEWS OF CZESTOCHOWA UNIVERSITY OF TECHNOLOGY, page 67, 2018.

[12] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. arXiv preprint arXiv:1310.4546, 2013.

[13] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pages 1532–1543, 2014.

[14] Sofia Visa, Brian Ramsay, Anca L Ralescu, and Esther Van Der Knaap. Confusion matrix-based feature selection. MAICS, 710:120–127, 2011.

[15] David MW Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. arXiv preprint arXiv:2010.16061, 2020.

[16] Aized Amin Soofi and Arshad Awan. Classification techniques in machine learning: applications and issues. Journal of Basic and Applied Sciences, 13:459–465, 2017.

[17] Kevin Gurney. An introduction to neural networks. CRC press, 1997.

[18] Marius-Constantin Popescu, Valentina E Balas, Liliana Perescu-Popescu, and Nikos Mastorakis. Multilayer perceptron and neural networks. WSEAS Transactions on Circuits and Systems, 8(7):579–588, 2009.

[19] Mikael Boden. A guide to recurrent neural networks and backpropagation. the Dallas project, 2002.

[20] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. arXiv preprint arXiv:1804.07461, 2018.

[21] Peipei Xia, Li Zhang, and Fanzhang Li. Learning similarity with cosine similarity ensemble. Information Sciences, 307:39–52, 2015.

[22] Prakash M Nadkarni, Lucila Ohno-Machado, and Wendy W Chapman. Natural language processing: an introduction. Journal of the American Medical Informatics Association, 18(5):544–551, 2011.

[23] Bertin Klein, Stefan Agne, and Andreas Dengel. On benchmarking of invoice analysis systems. In International Workshop on Document Analysis Systems, pages 312–323. Springer, 2006.

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in neural information processing systems, pages 5998–6008, 2017.