# Detektering af diagnoser i EKG-notater med NLP

---

Martin S. Jensen

*Juni, 2019*

Aalborg Universitet

AALBORG UNIVERSITY
STUDENT REPORT

School of Medicine and Health
Biomedical Engineering and Informatics
19gr10420

ST10

# Detektering af diagnoser i EKG-notater med NLP

Martin S. Jensen

Juni, 2019

**Martin S. Jensen**

*Detektering af diagnoser i EKG-notater med NLP*

ST10, Juni, 2019

AALBORG UNIVERSITET
STUDENTERRAPPORT

**SCHOOL OF MEDICINE AND HEALTH**
Sundhedsteknologi
Frederik Bajers vej 7E
9220 Aalborg Ø
Phone: 9940 8752
smh.aau.dk

**TITEL:**

Detektering af diagnoser i EKG-notater

**SEMESTER:**

P10

**PROJEKTPERIODE:**

Februar - Juni 2019

**PROJEKTGRUPPE**

19gr10420

**VEJLEDERE:**

Kirstine R. Gøeg (hovedvejleder)
Mathias B. Lanng (Bivejleder)

**FORFATTER:**

Martin S. Jensen

**SYNOPSIS**

Electronic patient records contain a wide range of diverse information, which has significance in regards to patient follow-up and further treatment. These records consist of vast amounts of data, much of which is in free text form, which can be time consuming for the clinicians to read tho through, in search of important diagnoses. It is therefore necessary to develop documentation systems that support these large amounts of data. If this data can be structured, the benefits include time conservation for the clinician, and quality of treatment for the patient. The goal of this project was to investigate how neural networks can be used to detect diagnoses and their context in ECG notes, when taking into account that the amount of annotated data for clinical NLP is limited. Two LSTM networks were used for the detection of candidate concepts and context classification, respectively. In context classification, on a subset of ECG notes, an F1 score of 0.92 was measured. The system was also tested on the shARe dataset, where the concepts were linked to a SNOMED CT code. A precision, recall and strict F-score of 0.98, 0.98 and 0.98 was obtained, respectively. The model was found to be capable of detecting diagnoses with a high level of confidence.

# Forord

Denne rapport er en 10. semester speciale rapport af Martin Sung Jensen, gruppe 19gr10420 på civilingeniøruddannelsen Biomedical Engineering and Informatics på Aalborg Universitet. Projektet er udarbejdet i perioden fra den 4. Februar til og med den 6. Juni 2019. Projektet omhandler udviklingen af et system der kan detektere diagnoser i EKG-fritekstnotater, og dermed effektivisere arbejdsprocessen herved. En stor tak skal rettes til Kirstine Rosenbeck Gøeg og Mathias Buus Lanng for kyndig vejledning gennem projektet.

*Martin Sung Jensen*

# Læsevejledning

I denne rapport benyttes Vancouver metoden til referering. Dette betyder at en henvisning til en kilde, foregår ved at indsætte kildens nummer i en firkantet parentes, f.eks. [1]. Hvis kilden er angivet til venstre for "." betyder dette at kilden kun gælder for den sætning, som punktummet afslutter. Hvis henvisningen er på højre side af "." vil kilden gælde for hele afsnittet. Litteraturlisten findes bagest i rapporten. Derudover er alle figurer og tabeller nummereret, og der findes to lister med disse bagest i rapporten.

Eksempler fra kildekoden kan ses i appendix C og hele kildekoden fra dette projekt er frit tilgængelig på Github på følgende link: *https://github.com/xraycat123/10sem*

# Indhold

# 1 Introduktion

Elektroniske patientjournaler indeholder en bred vifte af forskelligartet information. En stor del af denne information består af lavt strukturerede fritekstnotater som f.eks. EKG-beskrivelser, røntgenbeskrivelser og epikriser. Disse er med til at redegøre for patienternes sygehistorie og fund, samt danne basis for videre planlægning af patientforløb. En ulempe ved den lavt strukturerede data, er at den ikke umiddelbart er ligeså velegnet til primære og sekundære formål som f.eks. klinisk beslutningsstøtte, monitorering, facilitering af patientoverblik eller dataanalyse, som det mere velorganiserede, højt-strukturerede data. Dette skyldes at den lavt strukturerede data er mere vanskelig at overskue. [25] Der findes dog potentiale i den ustrukturerede data, som typisk er mere nuanceret, eftersom disse nuancer kan bidrage til en mere kvalitativ beslutningsstøtte. For at højne kvaliteten af den lavt strukturerede fritekst, kan denne struktureres ved at annotere vigtige begreber som f.eks. kliniske fund, anatomiske begreber eller medicin. For at kunne udtrykke begreberne på en præcis og konsistent måde, kan begreber sammenkobles med en sundhedsterminologi som SNOMED CT. Eftersom der findes meget store mængder af fritekst-baseret data, vil det være meget tidskonsumerende at gennemgå manuelt.Derfor kan det være hensigtsmæssigt at anvende Natural Language Processing (NLP) teknikker, til automatisk detektion af klinisk vigtige begreber fra notaterne. At kunne udføre denne detektion automatisk er dog ikke en triviel opgave, da klinisk tekst ofte indeholder mange forkortelser, stavefejl og tvetydige begreber, hvilket kan vanskeliggøre arbejdet for en algoritme. [39]

Hvis termerne i fritekstnotaterne er bedre strukturerede vil man, som nævnt i det ovenstående, f.eks. kunne bruge disse til klinisk beslutningsstøtte, hvis systemet er tilstrækkeligt præcist og stabilt. Beslutningsstøtten kunne f.eks. være med henblik på risikohåndtering ift. medicin eller til brug af prædiktive algoritmer. F.eks. ved kardiologiske notater vil en mulighed være, at man kan danne sig et bedre historisk overblik over de mulige kardiologiske ændringer, f.eks. som følge af indtaget medicin, som i nogle tilfælde først viser sig efter lang tid. Af sekundære formål kan nævnes administrative aspekter, som bedre overvågning af hospitalsaktivitet, og forskningsrelaterede områder som f.eks. epidemiologiske undersøgelser. Her kunne kliniske EKG-fund relateres til bestemte begivenheder, som f.eks. før/efter indsættelse af pacemaker. [25, 12] Der er altså potentielt en række fordele forbundet med udviklingen af et system, der kan skabe struktur i lavt struktureret tekst.

# Initierende problemstilling

*Hvilke redskaber benyttes til strukturering af kliniske termer i klinisk fritekst, og hvilke muligheder, udfordringer og fremtidsperspektiver er der ift. at kunne strukturere EKG-notater?*

# 2 Problemanalyse

*Dette kapitel søger at klarlægge hvilke redskaber der, på nuværende tidspunkt, benyttes til detektering af termer i kliniske fritekstnotater. Indledningsvist analyseres formålet med EKG-notater, og deres funktionalitet. Derudover vil der, i denne forbindelse, argumenteres for muligheder og udfordringer ved EKG-notater. Slutteligt undersøges hvilke metoder og redskaber, der findes til detektering af kliniske begreber i fritekst, og hvilke muligheder og begrænsninger disse metoder indeholder. Dette vil ende ud i en problemformulering, der vil danne basis for dette projekt.*

## 2.1 EKG-notaters formål og funktionalitet

*Indledningsvist beskrives og evalueres EKG-notaters funktionalitet, indhold og problematikker, der knytter sig disse. Efterfølgende analyseres muligheder og udfordringer, der kan opstå ved dokumentation i relation til EKG-dokumenter.*

Da hjerte-kar-sygdomme består af en række forskellige sygdomme, findes et utal af diagnosticerings- og behandlingsmuligheder. Her kan bl.a. nævnes ekkokardiografi, blodprøver, angiografi og EKG. EKG'et er den hyppigst benyttede diagnosticeringsmetode, hos patienter ved mistanke om en hjertesygdom [3]. Ikke alle hjertesygdomme afspejles i EKG'et, men mange gør, og eftersom det samtidig er en hurtig og simpel metode, er EKG'et en af de mest benyttede metoder [51]. EKG udføres kun ved mistanke om hjerteproblemer, og bliver derfor ikke benyttet som screeningsmetode på umiddelbart raske personer. Dette skyldes, at positive svar i visse tilfælde kan føre til invasiv diagnosticering for verificering af EKG-resultatet [9] En af de hyppigste dødsårsager, både nationalt og internationalt, er hjerte-kar-sygdomme, og antallet af personer, der lider af hjerte-kar-sygdomme er stadigt stigende. Vi lever generelt længere, og der findes stadigt bedre behandlingsmuligheder for denne gruppe af patienter, hvorfor de typisk lever længere med deres sygdomme. En naturlig konsekvens af dette øgede antal personer med hjerte-kar-sygdomme, og deres forlængede levetid med sygdommene er, at mængden af dokumentation vokser. F.eks. udføres der i USA omkring 300 millioner EKG'er hvert år [37]. Det er derfor nødvendigt, at udvikle dokumenteringssystemer der understøtter denne enorme mængde data, hvoraf meget er på fritekstform.

### 2.1.1 Procedure i forbindelse med optagelse og håndtering af EKG

I dette projekt defineres EKG-dokumenter som EKG-måling tilknyttet metadata, dvs. signaler, annoteringer, tekst osv. EKG-notater defineres som fritekstdelen af EKG-dokumentet. En typisk procedure for opsamling og håndtering af EKG er som følger: En sygeplejerske, eller lignende, sætter elektroder på patienten. Derefter optages EKG-målingen, og signalet præprocesseres og lagres. Når målingen er blevet lagret, visualiseres den, og en kardiolog kan inspicere målingen. Efter kardiologen har undersøgt EKG'et, kan vedkommende dokumentere observationerne, som derefter lagres.[2] Processen er vist på figur 2.1


**Fig. 2.1.:** Illustration viser arbejdsprocessen omkring EKG-målinger.

Resultaterne fra EKG-undersøgelserne er ofte en blanding af strukturerede og ikke-strukturerede resultater. De strukturerede dele af EKG-noterne kan f.eks. være de forskellige intervaller i EKG'et, mens der i den ustrukturerede del findes tolkninger af de konkrete målinger. Sidstnævnte har til formål at kommunikere diagnoser, fund og mistanker, hvilket f.eks. kan benyttes i forhold til håndteringen af medicin [10]. Der findes flere standarder, i forhold til lagring og udveksling af EKG-dokumenter, som har til formål at øge interoperabiliteten i elektroniske patientjournaler. De mest udbredte EKG-dokument-standarder er SCP-ECG, som siden 2005 har været en officiel standard i Europa [54], og HL7 aECG [4]. På trods af forskellige karakteristika, gæl-

der det for begge at det er muligt at tilknytte fritekst-notater i EKG-dokumenterne. [15, 4]

Som nævnt ovenfor, indeholder EKG-notater grundlæggende to slags information. Den ene er den morfologiske beskrivelse af EKG'et, og den anden er en fortolkning af den morfologiske ændring. "Forlænget Q-T interval" er et eksempel på en beskrivelse af morfologien i EKG'et, mens "myokardieinfarkt" er et eksempel på en fortolkning af et morfologisk fund. [12] I dag er EKG-maskiner udstyret med automatiske algoritmer. Output fra computerbaserede metoder, som beskriver f.eks. intervaller i EKG'et, betragtes som den gyldne standard i forhold til aflæsning af konkret data fra EKG'et [12]. Disse outputs lagres i EKG-dokumentets strukturerede del. Algoritmer der opererer på samme kvalitetsmæssige niveau som kardiologer eksisterer, men oftest foretrækkes kardiologen til at fortolke morfologiske fund og forskellige arytmier. Ofte vil man altså anvende en kardiolog til at afkode de aflæsninger man har opnået via den computerbaserede metode, og det er her den lavt strukturerede fritekst opstår.

### 2.1.2 Problemstillinger forbundet med kliniske dokumenteringsredskaber i EKG-kontekst

Klinikere benytter en signifikant del af deres arbejdstid på at arbejde i EPJ-systemer, og særligt dokumentation optager en stor del af arbejdstiden. I et amerikansk studie af Sinsky m.fl. [52] blev det fundet, at for hver time klinikerne benyttede på direkte patient-kontakt, brugte de to timer på at arbejde i EPJ. Studiet omfattede 57 læger fra fire specialer: almen medicin, intern medicin, ortopædkirugi og kardiologi. Det foreslås i studiet, at det kunne være hensigtsmæssigt at benytte redskaber til dokumentationssupport for at nedsætte arbejdstiden i EPJ, og samtidig øge tiden med den enkelte patient. [52] Generaliserbarheden af Sinsky m.fl. [52] kan dog diskuteres, da lokale forhold, erfaring, og forskellige IT-systemer i sådanne studier vil have indflydelse på dokumentationsbyrden.

En særlig udfordring for klinikerene, er den store mængde ustruktureret data. Ustrukturerede, kliniske narrativer er en af de største datakilder i sundhedsvæsenet [25], og i modsætning til struktureret data giver den ustrukturerede fritekst i højere grad mulighed for at udtrykke nuancer, overvejelser og symptomer. [48] En udfordring ved ustruktureret fritekst er dog den heterogene struktur, som medfører, at det ofte er mere omstændeligt at analysere denne type data end de strukturerede data. Dette skyldes blandt andet, at klinisk fritekst ofte indeholder stavefejl, forkortelser og tvetydighed, og er samtidig ofte kontekstafhængig [39]. Der findes et stort potentiale i den ustrukturerede data, men for at kunne udnytte disse data mere hensigtsmæssigt, skal der tages højde for den mere komplekse analyseprocess, hvorfor det her

vil være relevant at inddrage Natural Language Processing (NLP), som omhandler behandling og processering af sprog, ved hjælp af computere.

## 2.2 Fra fritekst til struktur med information extraction

*I dette afsnit beskrives hvilke muligheder der findes, i forhold til at strukturere klinisk fritekst. Først gives en beskrivelse af hvad information extraction er, og dernæst belyses de forskellige paradigmer inden for information extraction. Efterfølgende beskrives og evalueres forskellige metoder til klinisk information extraction, og slutteligt belyses problematikken ved prædiktive algoritmer, den mængde data disse algoritmer ofte er afhængige af, samt deres potentiale i forhold til EKG-notater.*

Information extraction (IE) handler om at finde bestemt information i tekst, og derefter skabe en struktureret repræsentation af den fundne information [44]. Ofte er konteksten som den udtrukne information befinder sig i, af væsentlig betydning. Konteksten kan f.eks. være hvorvidt en entitet er negeret, hvilken temporal kontekst entiteten har, eller hvilket subjekt denne tilhører [39]. Det er essentielt at tage højde for dette, da en forkert vurdering af konteksten, i yderste konsekvens kan have afgørende betydning for patientsikkerheden [5]. At identificere denne kontekst kan umiddelbart virke let for et menneske, men for en computer er dette ikke en triviel opgave. I klinisk øjemed er det ofte ikke tilstrækkeligt, udelukkende at kunne identificere entiteter, som f.eks. diagnoser, medicin osv. Det er tværtimod nødvendigt samtidig at normalisere de fundne entiteter, forstået på den måde at de oversættes til en standardiseret terminologi [48] som f.eks. SNOMED CT eller UMLS. Denne form for begrebsnormalisering er også med til at sikre systeminteroperabilitet for kliniske IT-systemer [39]. Oftest er fremgangsmetoden således at der først lokaliseres kandidatbegreber, tilhørende en begrebsklasse, som er af mulig interesse i forhold til videre processering. Et eksempel på lokalisering af et kandidatbegreb er illustreret i tabel 2.1. Efterfølgende benyttes eksempelvis opslagsværk til at sammenkoble disse til unikke koder, hvilket medfører, at begrebet "atrial fibrillation"mappes til `49436004 | Atrial fibrillation (disorder) |` Efter en entitet er blevet sammenkoblet med en unik kode, kan denne benyttes til videre processering. F.eks. kan man forbedre søgbarheden for kliniske notater, da man via denne metode vil kunne søge på nøgleord, i stedet for at gennemgå en masse overflødig tekst [25, 39]. Det er også muligt at benytte de fundne begreber som input til prædiktive algoritmer, som f.eks. bruges i forbindelse med klassificering af patienter [21].

En af de vigtigste og mest grundlæggende opgaver indenfor IE, er at finde og klassificere entiteter i en given tekst. Dette kaldes Named Entity Recognition (NER), og er illustreret på figur 2.1

| The | rhythm | appears | to | be | atrial | fibrilation |
|---|---|---|---|---|---|---|
| O | O | O | O | O | B-disorder | I-disorder |

**Tab. 2.1.:** I eksemplet benyttes IOB tagging formatet, hvilket er et af de mest benyttede formater til Named Entity Recognition. O står for “outside”, B står for ”begin” og I står for ”inside.

For at finde entiteter i tekst benyttes algoritmer, og i NLP findes to paradigmer: regelbaserede og statistiske metoder. De to skal ikke ses som værende gensidigt udelukkende metoder, men derimod som metoder der kan supplere hinanden. De statistiske metoder har dog fået megen opmærksom de seneste år. Inden for de traditionelle ikke-kliniske NLP metoder dominerer statistiske metoder, da de har vist at være mere præcise end de regelbaserede. Især neurale netværk har vist sig at være bedre end andre traditionelle machine learning metoder, indenfor en lang række NLP-områder [62].

### 2.2.1 Regelbaserede og machine learning metoder til information extraction

De regelbaserede algoritmer er, som navnet indikerer, algoritmer der benytter regler og regular expressions, og ofte er disse metoder stærkt afhængige af specialiseret domæneviden. På trods af en ofte bedre ydeevne ved brug af statistiske metoder, er regelbaserede algoritmer stadig hyppigt brugt i anvendt klinisk NLP. Dog bliver systemer, der udelukkende baserer sig på regelbaserede systemer, i forskningen betragtet som forældede [60]. På trods af den faldende popularitet og overvejende lavere præcision har regelbaserede algoritmer den klare fordel, at de nemt kan ændres ved uønsket adfærd. [25] I de typisk anvendte frameworks cTakes og medLEE benyttes disse ofte i kombination med statistiske metoder [31].

I litteraturen er machine learning og hybridmodeller ofte de algoritmer der har den højeste præcision på standard NER-datasæts, og også på kliniske datasæt [60]. Ulempen ved machine learning modeller er, at det kan være svært at tage højde for systematisk bias i modellerne [25]. Den mest benyttede form for machine learning er superviseret læring, hvormed der benyttes en algoritme som kan tillæres en funktion, der kan mappe et input til output ved hjælp af labelled data. Sagt på en anden måde, så har de eksempler man præsenterer algoritmen for et label; f.eks. er hvert ord tilknyttet et mærkat, som indikerer hvilken klasse ordet tilhører.

### 2.2.2 Oversigt over Information Extraction studier

Der blev fundet ét studie fra 2005 af Denny og Anderson Spickard [11], som havde undersøgt klinisk entity extraction i relation til EKG-dokumenter. De benyttede algoritmen "Knowledge map"[13], hvilket bl.a. var baseret på nominal frase-identificering og string matching. Algoritmen havde en F1-score på 0.91, men forfatterne af studiet påpeger dog, at dette ikke umiddelbart kan overføres til andre forhold, idet mange sætninger i notaterne bestod af standardiserede sætninger. Det var her forventet, at jo mere tekst der var indskrevet direkte af kardiologen, jo lavere præcision ville algoritmen have, da dette øger andelen af stavefejl og ikke-standardiserede forkortelser. Majoriteten af fejldetektionerne i studiet skyldes samme problematik. Derudover benyttede Denny og Anderson Spickard [11] et datasæt, der ikke er tilgængeligt for yderligere forskning, hvilket besværliggør sammenligning med andre metoder. Grundet fåtallet af studier, som tager udgangspunkt i EKG-notater alene, blev det undersøgte domæne udvidet til også at inkludere studier, som omhandlede generel klinisk entity extraction.

Der blev foretaget en litteratursøgning der inkluderede studier fra år 2005 til 2019, hvilket resulterede i otte artikler med følgende inklusionskriterier:

1. Engelsksprogede datasæt

2. Datasæt baserede på kliniske journaler

3. Baseret på åbne datasæt

Studier omhandlende relation extraction er ikke medtaget, selvom dette hører under information extraction. Alle inkluderede studier har været en del af en konkurrence, og det er valgt at kun den vindende løsning i konkurrencen, samt eventuelle fremtidige forbedringer på det samme datasæt, vil indgå. De inkluderede studier er vist i tabel 2.2. Blandt de fundne studier havde fire af dem inkluderet EKG-notater i træningssættet. Dog var disse ikke inkluderet i testsættene.

| Dataset | Årstal | Hold | Algoritme | F1(S) | F1(R) |
|---|---|---|---|---|---|
| CLEF 1a* [46] | 2013 | UTHealthCCB [57] | CRF + SVM + VSM | 0.75 | 0.87 |
| CLEF 1b* [46] | 2013 | NCB1.2 [57] | CRF + Dnorm | 0.59 | 0.86 |
| SE+ task 7* [45] | 2014 | UTH CCB [64] | CRF + SVM | 0.741 | 0.87 |
| SE+ task 14* [16] | 2015 | ezDI [43] | CRF+SVM | 0.757 | 0.79 |
| i2b2 [58] | 2010 | Martin et al. [35] | HMM +cTakes + Metamap+ ConText | 0.852 | 0.92 |
| i2b2 [58] | 2017 | Liu et al. [33] | LSTM | 0.86 | - |
| i2b2 [55] | 2017 | Liu et al. [33] | LSTM+CRF | 0.923 | - |
| i2b2 [55] | 2012 | Sun et al. [56] | CRF | 0.920 | 0.86 |

**Tab. 2.2.:** Inkluderede studier. Der er inkluderet to studier til hvert i2b2 datasæt. Dette er som følge af at artikler med en højere rapporteret performance er blevet udgivet efter konkurrencens afslutning. F1(S) og F1(R), er en forkortelse for henholdsvis strict F1 og relaxed F1. *inkluderer EKG-notater i træningssættet. +SemEval

Overordnet blev det observeret, at flertallet af disse algoritmer er baseret på conditional random fields algoritmen (CRF), samt support vector machine (SVM). Dette stemmer overens med et review af Wang m.fl. [60], der undersøgte de mest benyttede algoritmer og frameworks, og hvor der inddrages studier fra 2008-2016. Det blev fundet, at de mest benyttede algoritmer var Support Vector Machine, Logistisk regression og CRF.

Disse algoritmer er afhængige af megen feature engineering, hvilket også er tilfældet for algoritmerne i de inkluderede studier. Derudover var flere af løsningerne i de inkluderede studier ofte specielt konstruerede til den givne opgave, og det må formodes at disse er dårligt rustede til at generalisere. Selvom algoritmerne i studierne ikke var ens, var fremgangsmetoden i forhold til at koble en kode til et kandidatbegreb tæt relateret. Udover en høj præcision, udmærker neural netværk-baserede løsninger, som LSTM, sig ved at de ikke ligeså ofte er afhængige af manuel feature engineering som de andre algoritmer.

### 2.2.3 Valg af information extraction metode og udfordringer

I dag har neurale netværk opnået State-of-the-art performance indenfor mange områder i machine learning – også indenfor NLP. Heriblandt kan nævnes talegenkendelse, named entity recognition, maskinoversættelse og tekstklassifikation. På disse områder har neurale netværk vist sig at være andre metoder overlegen, så længe der er nok data. Tendensen har dog ikke været lige så klar inden for klinisk named entity recognition, da man ikke har lige så store og åbne datasæt til rådighed. Der er afholdt flere konkurrencer i klinisk entity recognition, som nævnt i afsnit 2.2.2. Efterfølgende studier har vist, at man med mere moderne metoder kunne opnå ligeså god eller

bedre præcision med neural-baserede metoder, der ikke i samme grad afhænger af feature engineering som f.eks. SVM eller CRF-modeller. Dog afhænger de ofte af større datasæt end andre gængse metoder [33, 22, 34] . Dette understreges også af Wang m.fl. [60], der ikke har fundet nogle løsninger baserede på neurale netværk. Wang et. al argumenterer for, at denne mangel på løsninger kan skyldes manglen på store åbne datasæt, der er frit tilgængelige til forskning [60]. Dette er problematisk i forhold til klinisk data, da høje krav til håndtering af private og fortrolige oplysninger medfører, at der ikke er store åbne annoterede datasæt tilgængelige [28]. Derudover kræves der i det kliniske domæne, en ekspertviden for at kunne identificere og annotere tekst, hvilket der kan være væsentlige omkostninger forbundet med. Der findes dog åbne ressourcer, som eksempelvis MIMIC-III databasen, der indeholder data fra over 40.000 patienter. Journalerne fra MIMIC-III databasen er dog ikke annoteret. Forskellige metoder forsøger at håndtere denne problemstilling, f.eks. ved hjælp af active learning. Formålet med active learning er at udvælge de mest informative eksempler, i stedet for at udvælge eksempler tilfældigt, hvilket er normal praksis ved superviseret læring. Idéen er, at den prædiktive algoritme kan performe bedre, med mindre data end der normalt kræves. Dermed er der et stort potentiale ift. at minimere mængden af tid og arbejdskraft, gennem konstruering af machine learning modeller. [28]

## 2.3 Problemformulering

I problemanalysen blev problematikken, i forhold til at journaler ikke umiddelbart lader sig strukturere, belyst. Denne udfordring medfører at den detaljerige information der findes i notaterne, på nuværende tidspunkt ikke utiliseres optimalt. Dette er særligt aktuelt set i lyset at den store, og voksende, mængde EKG-notater der genereres i sundhedsvæsenet. Hvis fritekst-notaterne kan struktureres ved hjælp af IE, kan dette forbedre søgbarheden i dokumenterne. Dette kan ultimativt være med til at nedbringe den tid, som klinikerene benytter til at danne sig et overblik over patienternes historik, og derved frigive tid, som klinikeren så kan bruge på andre opgaver, som f.eks. direkte tid med patienten. I problemanalysen blev det sandsynliggjort at datadrevne metoder, som neurale netværk, udgør et uudnyttet potentiale. En udfordring ved at benytte neurale netværk til IE er dog, at det er nødvendigt at tage højde for behovet for den større datamængde, disse algoritmer ofte kræver. Derudover fandtes ingen studier, der har belyst om neurale netværk kan benyttes til IE af EKG-noter.

Derfor vil en løsning kunne være at konstruere et system, der løbende kan indsamle annoteret data, som kan benyttes til at forbedre modellen efter implementering. Dette kan gøres ved at klinikerene inddrages i annoteringsprocessen, så klinikerene

kan verificere eller tilrette modellens prædiktion, efter klinikeren har skrevet et EKG-notat. Dermed er følgende problemformulering opstillet:

*Hvordan kan neurale netværk anvendes til at detektere diagnoser og deres kontekst i EKG-notater, når der skal tages højde for at mængden af annoteret data til klinisk NLP er begrænset?*

# 3 Metode

## 3.1 Systemkoncept

*I dette afsnit præsenteres konceptet for det færdige system. Formålet med systemet er at strukturere EKG-notaterne, og derved gøre det lettere for klinikeren at få et overblik over en patients historik.*

Systemet skal fungere på en sådan måde at det løbende bliver bedre når klinikeren bruger det. Således at når klinikeren bekræfter eller retter en foreslået kode, forstås dette som ny annoteret tekst, der kan bruges til at træne modellen - hvilket gøres i systemets run-time. Der refereres til dette som den *inkrementelle model*. Konceptet vil fungere som illustreret på figur 3.1

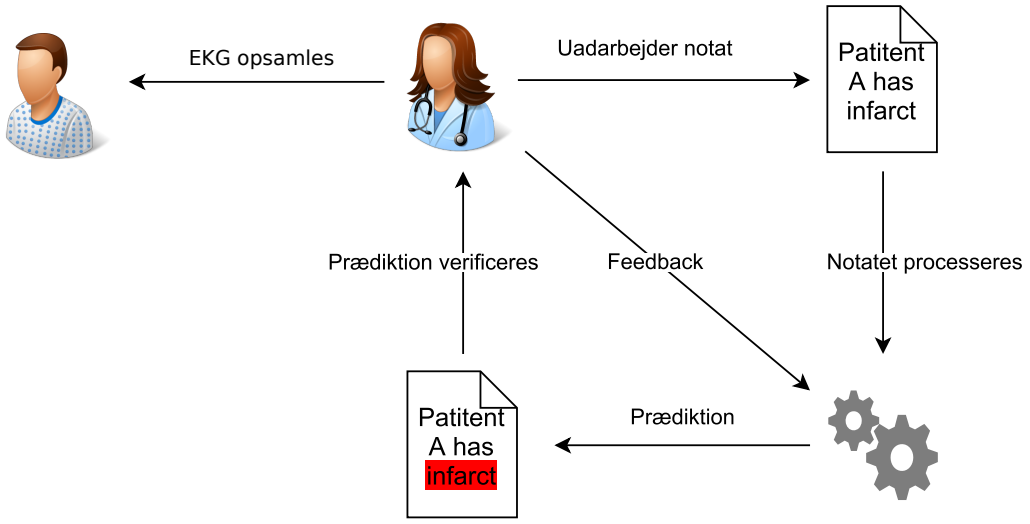


**Fig. 3.1.:** Illustration af systemkonceptet. Klinikeren kan udarbejde et notat, og derefter kan en algoritme detektere diagnoser. Klinikeren har derefter mulighed for at verificere eller afvise en prædiktion.

Når en patient har fået foretaget en måling, analyserer klinikeren EKG-målingen og nedskriver et notat. Derefter bliver notatet indført i *tekstgenkendelsessmodulet*, og modellen prædikterer hvilke diagnoser der er nedskrevet i notatet, samt i hvilken kontekst disse optræder. Klinikeren har mulighed for at bekræfte prædiktionen, men kan også tilrette svaret. F.eks. kan det angives at en af de fundne diagnoser er forkert,

eller der kan rettes på en forkert kontekst-værdi. Verificerede prædiktioner lagres i en database, og disse benyttes til at gentræne modellen.

Det er blevet fravalgt at opdatere modellen efter hver enkelt prædiktion, da der dermed opstår en risiko for, at modellen ikke optimeres hensigtsmæssigt. Dette kunne f.eks. være, hvis modellen modtager mange eksempler i træk der ligner hinanden, da dette vil kunne medføre at der opstår bias i modellen. Samtidig vil det være besværligt og unødigt beregningstungt, hvis modellen skulle gentrænes med alle dataeksempler efter hver prædiktion. I stedet vil modellen blive gentrænet efter behov, f.eks. efter en fastlagt periode, når der er blevet lagret x antal målinger, eller hvis modellens prædiktioner overskrider en fastlagt tærskelværdi for procentvis forkerte prædiktioner. Hvor meget ny data der skal til for at modellen vil øge dens performance, vil afhænge af dataens kvalitet, mængden af information som modellen kan udnytte, og hvordan modellen allerede performer. En model der i forvejen performer højt, vil højst sandsynligt have sværere ved at frembringe de samme fremskridt, som en model der ikke er trænet på megen data, og som derfor ikke havde en ligeså høj performance til at begynde med. Det kan heller ikke forventes at modellen vil forbedre sig synderligt, hvis den bliver udsat for mange af de samme slags dataeksempler. Hvis modellen udsættes for få eksempler, der ikke minder om noget der er set før, må der derimod forventes større fremskridt. Dette er bl.a. beskrevet i et studie af Shen m.fl. [50], hvor forskellige annoteringsmetoder og deres effektivitet blev beskrevet. I dette studie testede man forskellige annoteringsstrategier på et anerkendt NER benchmark datasæt (OntoNotes 5.0). Studiet viste at ved at benytte forskellige active learning-strategier, stabiliserede F1-scoren sig omkring 40.000 annoterede ord, og performance blev kun øget med under 2 procentpoint(op til 80%) uanset hvor mange annoterede eksempler, der blev trænet på. Ved tilfældig sampling stabiliserede F1-scoren sig først efter 160.000 annoterede ord, og opnåede en tilsvarende performance, som ved active learning metoderne.

I dette system er hovedfokus på at detektere diagnoser. I systemet defineres diagnoser som det der svarer til *disorders*, defineret ud fra sundhedsterminologien SNOMED CT. En mere detaljeret vejledning er givet i appendix A. Som beskrevet i afsnit 2.2 vil det være nødvendigt, i et patientsikkerhedsmæssigt perspektiv, at detektere konteksten for de detekterede diagnoser. Derfor er der, i dette projekt, et krav om at en diagnose enten skal klassificeres som negeret, ubekræftet eller bekræftet. Derfor vil dette også blive en egenskab for systemet i dette projekt.

### 3.1.1 Eksempel på brugergrænseflade

I dette afsnit præsenteres en skitse af en brugergrænseflade for hvordan systemets notat modul kunne se ud. Denne er vist på figur 3.2 . I midten af brugergrænsefladen

er et tekstområde, hvor klinikerens fortolkning af EKG'et kan skrives ind. I den viste brugergrænseflade er det indskrevne notat blevet processeret af algoritmerne, der detekterer diagnoser og deres kontekst, og prædiktionen vises til brugeren. Brugeren kan verificere prædiktionen eller udsætte verificering i tilfælde af tidsmangel eller tvivlstilfælde. Brugeren har i tillæg mulighed for at tilrette detektionerne, hvis disse ikke er korrekte. I bunden af figuren er en tidslinje der viser patientens historik, som dermed kan give klinikeren et hurtigt overblik over patientens sygdomshistorik. I venstre side er det muligt at sortere i de registrerede diagnoser, og denne sortering kunne f.eks. være baseret på type, alfabetisk orden eller dato. I nederste venstre hjørne er de morfologiske værdier, som typisk findes som en struktureret datatype.

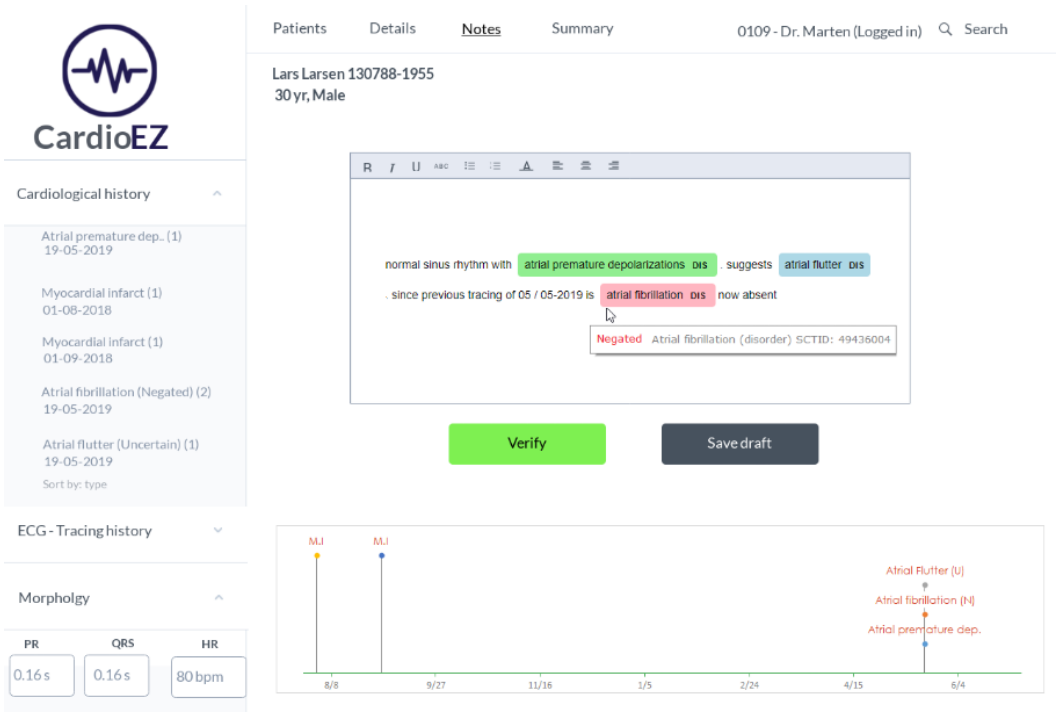


**Fig. 3.2.:** Figuren illustrerer en GUI for systemet. Brugergrænsefladen er modelleret ved hjælp af værktøjet MarvelApp [36].

## 3.2 Beskrivelse af data

Data benyttet i dette projekt stammer fra ShARe CLEF e-health challenge 2014 (ShARe) [41] og MIMIC-datasættet [26]. ShARe består af 300 ekspertannoterede noter, hvoraf 54 er EKG-notater. Notaterne fra ShARe CLEF er en del af MIMIC-datasættet, om end ShARe notaterne er annoterede. I datasættet er de termer der kan klassificeres som diagnoser i SNOMED CT hierarkiet, ekspertannoterede. De 54 notater indeholder 104 SNOMED CT koder.

MIMIC-III (Medical Information Mart for Intensive Care III) består af data fra 46.520 kritisk syge patienter fra Beth Israel Deaconess Medical Center i Boston, Massachusetts, i perioden 2008-2014. [26]. I MIMIC-III databasen er der 26 tabeller, som hver repræsenterer et afgrænset område af patientbehandlingen, f.eks. medicin, administrative oplysninger, parakliniske tests osv. En signifikant del af indlæggelserne i MIMIC-databasen er relateret til hjertesygdomme. Dette kommer f.eks. til udtryk ved at to af de tre hyppigste udskrivelses-koder, er relaterede til hjertesygdomme. Alle notater findes i en separat tabel, indeholdende alle noter fra MIMIC-databasen, hvor bl.a. udskrivelsesnotater, sygeplejernotater og EKG findes. I tabellen er alle unikke besøg udtrykt ved et unikt indlæggelses-ID. Der er 209.051 EKG-notater hvilket gør denne notattype til den tredje hyppigst forekommende i MIMIC-databasen. I EKG-notaterne er der 6.008.155 ord, hvoraf 14.629 er unikke ord. Ydermere er der 182.224 sætninger med en gennemsnitslængde på cirka 33 ord. Det betyder at dette korpus er relativt småt, sammenlignet med f.eks. udskrivningsnotaterne, som har ca. 78 tusinde unikke ord. Der skal dog tages højde for, at der er mange sammensatte ord og således mange kombinationstermer i medicinsk terminologi. Da noterne stammer fra rigtige patienter, har en grundig anonymiseringsprocess fundet sted, så patienterne ikke kan identificeres. Der er anvendt en proces der er i overensstemmelse med de amerikanske HIPPA-kriterier. Det betyder at alle datoer er forskudt, så datoer i MIMIC-III er kun gyldige i forhold til perioder for de enkelte patienter. I MIMIC-III er der benyttet regular expressions og opslagslister til at fjerne følsom informationer, hvormed telefonnumre, adresser, alder, navne osv. er anonymiseret.

Efter en manuel inspektion af 200 tilfældigt udvalgte EKG-noter blev det observeret, at den generelle struktur for noterne var, at den første sætning fortæller om den overordnede rytme for hjertet, f.eks. “sinus rhytm” (normal rytme) eller “atrial fibrillation”. Langt de fleste af de inspicerede EKG’er viste dog “sinus rhythm”. Dernæst fulgte ofte en beskrivelse af morfologien og en tolkning heraf. En anden observeret forskel var, at der ofte er forskel i opsætningen, f.eks. i form af linje skift eller ved tegnsætning. En udfordring ved noterne, er at de samme termer kan skrives på forskellige måder. Et eksempel på dette kunne være, at i nogle noter var

"myocardial infarct" nedskrevet som helt ord, mens det i andre blev forkortet som "m.i" . Derudover blev sporadiske stavefejl også observeret; bl.a. ordet "ischemia" kunne staves som "ischemii".

For at kunne udvikle algoritmerne, blev der konstrueret tre datasæt. På figur 3.3 er disse vist. De to datasæt med kandidatbegreber er identiske, bortset fra at datasæt B er annoteret med information om hvorvidt en diagnose er negeret, ubekræftet eller bekræftet. Grunden til at ShARe CLEF datasættet ikke benyttes til at detektere kontekst, skyldes at majoriteten af termerne ikke er negerede eller ubekræftede, og derfor vil et troværdig billede af en algoritmes performance ikke blive klarlagt.

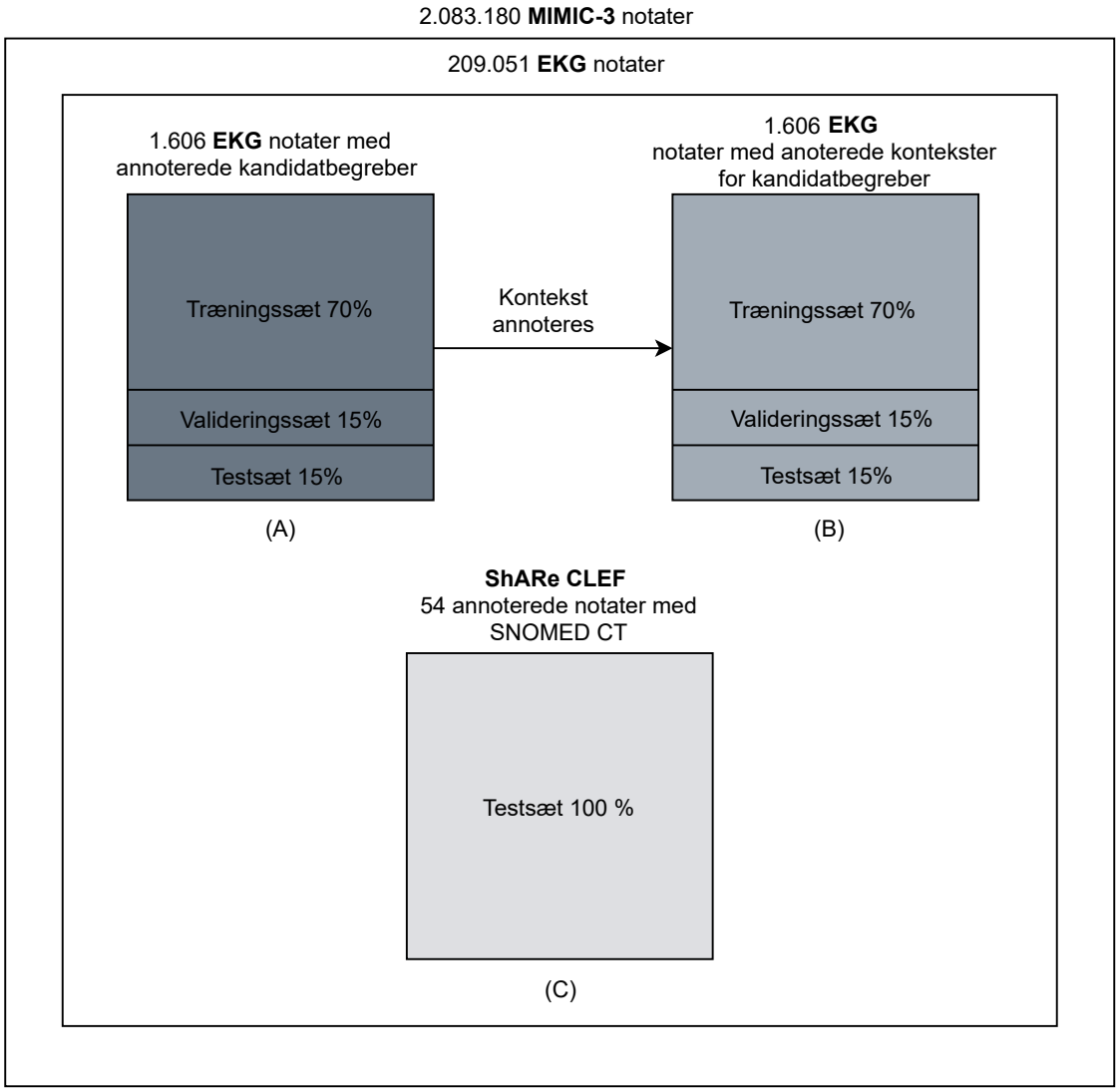


**Fig. 3.3.:** Figuren viser de tre forskellige datasæt fra MIMIC databasen. A og B indeholder annoterede kandidatbegreber samt konteksten for disse. C er det ekspertannoterede datasæt fra ShARe CLEF datasættet.

## 3.3 Udvikling af inkrementel model til Information Extraction

Disse trin forløber under run-time når brugeren benytter den inkrementelle model. Trinnene vil blive beskrevet i hvert sit afsnit, og afsnittene vil beskrive hvilken metode, der er blevet benyttet til at konstruere de enkelte dele. Pipelinen for den inkrementelle model er illustreret på figur 3.4 og vil blive yderligere uddybet i de efterfølgende afsnit.

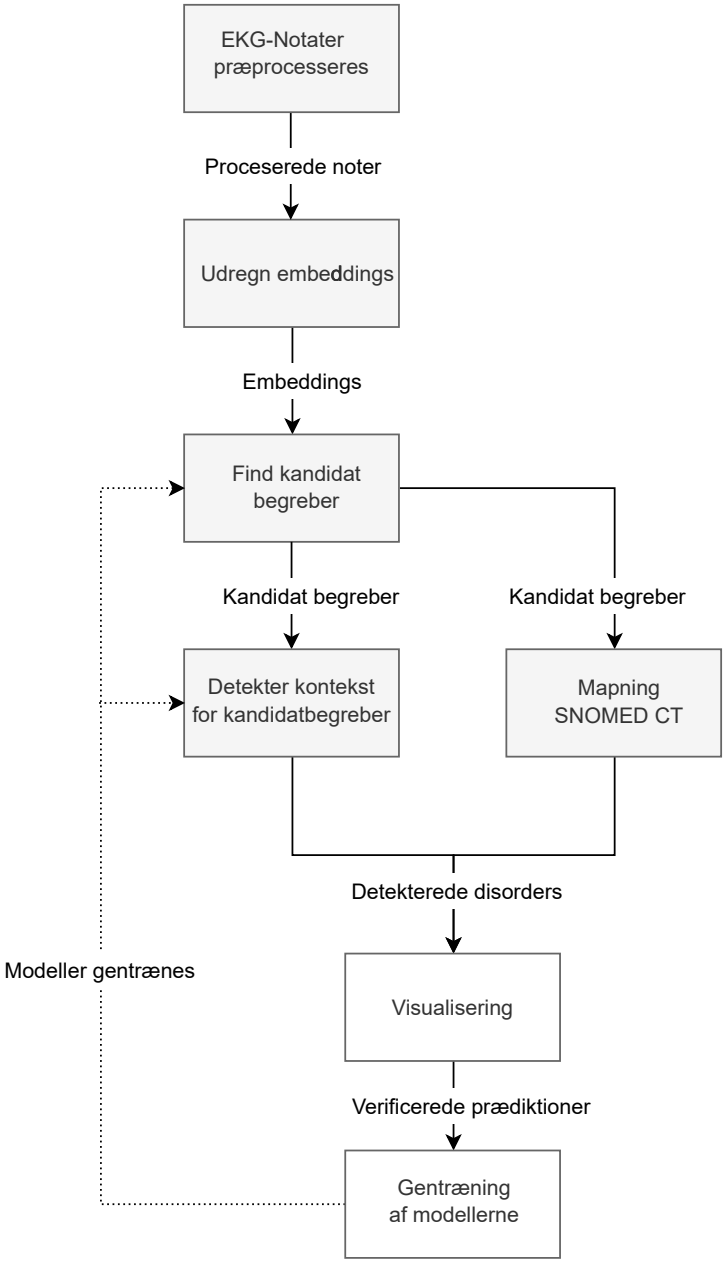


**Fig. 3.4.:** Figuren viser hvilke moduler der er tilknyttet den inkrementelle model. Visualisering og gentræningssystemet er en del af konceptet, men ikke en del af implementeringen.

Ovenstående pipeline skal ses som en række moduler, der udgør designet af systemet. Outputtet fra et modul vil blive benyttet i et eller flere moduler fremadrettet. Hvert af disse moduler kan udføres ved hjælp af forskellige algoritmer, som kan byttes ud med hinanden. Der vil ikke blive foretaget en evaluering af visualisering, eller af hvordan modellen gentrænes, da projektet har et andet fokus.

**EKG-notater præprocesseres**
Først præprocesseres EKG-notaterne, så de er lettere at bearbejde for algoritmerne. Dette gøres ved at fjerne unødig information, og derefter udføre tokenization.

**Udregn embeddings**
Dernæst laves word-embeddings, da disse er essentielle for at indlejre "mening" i ordene, for dermed effektivt at kunne træne machine learning modeller. Dette sker ved at træne på en stor mængde tekst fra MIMIC-databasen. To forskellige typer af embeddings blev benyttet: statiske og dynamiske typer.

**Find kandidatbegreber**
For at kunne detektere begreber i fritekstnotaterne, skal en NER-algoritme konstrueres. Denne er tidligere beskrevet i afsnit 2.2. Denne del af systemet vil detektere kandidater for diagnoser. For at kunne udvikle et neuralt netværk til denne opgave, skal der genereres megen annoteret data. For at imødekomme dette behov blev en active learning-strategi benyttet, til at sample effektivt, og udvælge de samples der kunne bidrage med mest information. Dette var nødvendigt, da der kun var 54 ekspertannoterede EKG-notater i ShARe datasættet. Til at udvælge eksempler, blev benyttet en LSTM-model. Denne model var identisk med den, der blev benyttet til NER-detektering.

**Detekter kontekst for kandidatbegreber**
Efter dette blev en klassifikationsmodel benyttet til at finde konteksten for de fundne diagnoser. Det vil sige, at det skal detekteres om en diagnose er negeret, usikker eller tilstede. Dette blev udført med endnu en neural LSTM-model, og denne blev konstrueret ved at benytte store mængder ikke-annoteret tekst til træning af modellen.

**Mapning til SNOMED CT**
Diagnoserne blev tilknyttet en unik SNOMED CT kode. Dette blev gjort ved brug af UMLS's web-api, der kan slå koder op baseret på tekstrepræsentation. Da det kræver en præcis søgestreng at slå SNOMED CT koder op, benyttes en Sieve-Based algoritme, der kan inddrage forskellige varianter af kandidatbegreberne.

**Visualisering**
En snitflade i en GUI blev foreslået, for at kunne illustrere, hvordan en brugssituation

kunne se ud for en kliniker, der skulle benytte systemet. Dette er en del af konceptet, men ikke en del af implementering.

**Gentræning af modellerne**

Gentræning af modellen vil, på et overordnet plan, fungere som beskrevet i afsnit 3.1. På et mere detaljeret plan vil en mulighed være, at fejldetekteringer kunne benyttes til at finde nye lignende eksempler, i ikke-annoteret fritekst, ved hjælp af forskelige metoder. Disse lignende eksempler kunne f.eks. findes vha. en regelbaseret algoritme, der kan gennemsøge en stor mængde tekst. Disse eksempler kunne derefter annoteres og anvendes til at gentræne modellen. Dermed vil man, ved hjælp af en klinikers feedback for et enkelt fejleksempel, potentielt være i stand til at forbedre modellen markant. Dette er en del af konceptet, men ikke en del af implementering.

### 3.3.1 Præprocessering af EKG-notaterne

Det er en nødvendighed at bearbejde EKG-notaterne, så disse kan benyttes som input til modellerne. For tekst gælder det generelt, at ord der starter med et stort forbogstav er egennavne. Det blev dog observeret i afsnit 3.2, at navne på sygdomme eller ligende ikke nødvendigvis blev stavet med stort forbogstav. Den eneste nogenlunde konsekvente brug af stort begyndelsesbogstav, var når ordet kom efter et punktum. Derfor blev alle bogstaver i de inkluderede EKG-noter konverteret til små bogstaver. Dette betyder, at denne egenskab ved ordene antages kun at indeholde minimal information, som machine learning algoritmerne kunne drage nytte af. Fordelen ved denne tilgang er at vokabulariummet gøres mindre, og dermed vil machine learning algoritmer have nemmere ved at lære. Alle informationer der kan være personhenførbare, blev anonymiseret som vist i afsnit 3.2. Disse blev identificeret vha. regular expressions og ændret til tegnet "_" sammenkædet med navnet på kategorien af det anonymiserede ord. Dette er blevet valgt, eftersom det formodes at de anonymiserede termer ikke vil bidrage med vigtig information. Ved en eventuel implementering af dette system vil disse anonymiserede ord ikke forekomme i samme form, og derfor vil håndtering af datoer, navne osv., skulle benytte andre regular expressions end der er benyttet i dette projekt. I EKG-notaterne blev det observeret, at noterne udelukkende omhandlede den patient, der blev målt på, hvorfor det blev vurderet, at alle navne kunne laves om til "_name", uden at det ville skabe forvirring. I andre tilfælde, som ved udskrivelses notater hvor patientens familiehistorik ofte nævnes, vil det være hensigtsmæssigt at have mere end én navnekategori.

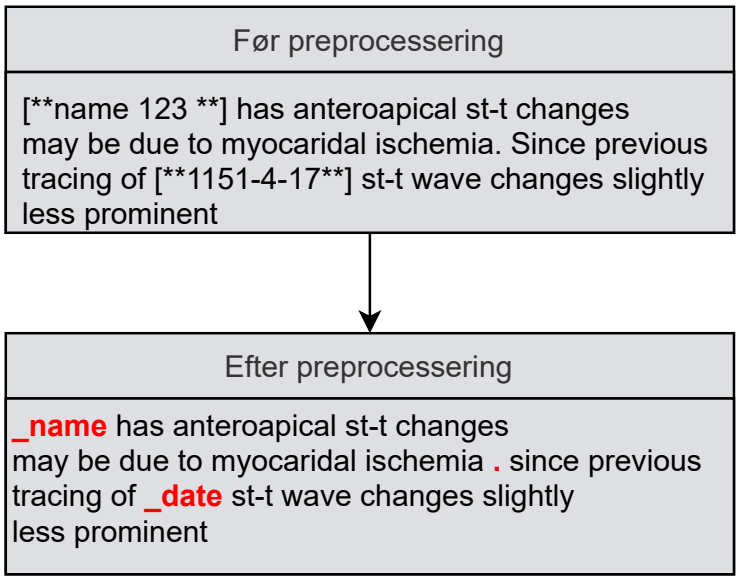


**Fig. 3.5.:** Figuren viser en tekst før og efter præprocessering. Det kan ses idet anonymiseringerne er lavet om. Derudover er punktummet blevet en selvstændig enhed.

Slutteligt blev alle ord tokenized, hvilket vil sige at de blev grupperet som brugbare semantiske enheder - altså at hvert ord kommer til at stå for sig selv. Her vil f.eks. tegnsætningstegn også blive adskilt fra ord. Et eksempel på dette er givet i figur 3.5. Til dette blev Python biblioteket Spacy benyttet til tokenization. Det blev fundet i 3.2 at EKG-noterne var relativt korte og ikke indeholdt flere afsnit, som det f.eks. ofte ses i epikriser, hvor der kan være afsnit vedrørende familie-historie, medicin, osv. Dette betød at hele tekststrengen i EKG-notaterne blev benyttet som input til modellerne.

### 3.3.2 Udregning af embeddings

En vigtig opgave indenfor NLP er at definere hvordan ordene repræsenteres som input til en model. Embeddings benyttes til mange opgaver indenfor NLP og har empirisk vist sig at være en nyttig repræsentation af ord. Embeddings leder ofte til en øgning i performance, hvis vektorerne benyttes til opgaver som f.eks. NER, tekstklassifikation og øvrige NLP-opgaver. At benytte wordembeddings er ikke en forudsætning for at benytte neurale netværk, men vil som sagt ofte øge performance for disse. Ved at repræsentere ord ved hjælp af embeddings, er det f.eks. muligt at repræsentere ord på en sådan måde, at nogle ord er tættere semantisk relaterede end andre, og dermed får ordene en slags "mening", i kraft af de ord der er en tæt relation til.

Grundlæggende kan embeddings opdeles i to typer: de statiske og dynamiske. De statiske har én bestemt vektor for et givet ord, mens en dynamisk vektor vil have en forskellig repræsentation afhængig af kontekst. I dette projekt benyttes word embeddings og contextualized string embeddings som input til LSTM-modellen, som beskrevet i afsnit 3.3.3, og til active learning, som beskrevet i samme afsnit. Til LSTM-modellen sammenkobles disse to vektorer til én vektor, som benyttes som input. Fremadrettet i rapporten vil contextualized string embeddings benævnes som character embeddings.

#### Word embeddings

I dette projekt benyttes word2vec-algoritmen til at konstruere word embeddings. I opgaver indenfor NLP benyttes ofte prætrænede vektorer, men dette er fravalgt i nærværende projekt, da det må formodes at sproget i notaterne adskiller sig for meget, i forhold til sproget i de korpusser som de prætrænede vektorer er baseret på. Word2Vec algoritmen er et to-lags, neuralt netværk, hvor der benyttes et stort tekstkorpus til at træne modellen. Her vil modellen prøve at gætte på om to ord befinder sig indenfor det samme kontekstvindue. Word2Vec modellen blev

implementeret ved hjælp af Gensim [47] biblioteket i Python. Standardindstillingerne fra Gensim er blevet benyttet for de fleste parametre. Dog blev antallet af negative samples ændret til 20, da Mikolov m.fl. [40] anbefaler, at der bør benyttes et højere antal negative samples for små datasæt(5-20), i forhold til større datasæt(1-5). Tallet fortæller noget om hvor mange negative eksempler modellen ekspooneres for i forhold til et enkelt kontekst ord. Mikolov m.fl. [40] definerer ikke hvad et "lille" datasæt er, men da forskellige ord i kendte Word2Vec implementeringer ofte tælles i millioner, betragtes det benyttede datasæt som værende et lille datasæt. Derudover er der blevet benyttet en lavere dimensionalitet på 100, da der opereres ud fra et relativt lille datasæt.

### Character embeddings

Som tidligere nævnt, benyttes character embeddings, som input til NER-modellen. Idet det hele fungerer på karakterniveau, er disse embeddings robuste over for stavefejl og over for ord, som ikke er specificeret i et vokabularium. De benyttede character embeddings adskiller sig fra traditionelle character embeddings [32] ved at være trænet usuperviseret på tekst som en character language model. Fordelen ved at benytte kontekstualiserede character embeddings kontra kontekstualiserede ord-embeddings er, at vokabulariummet er meget mindre. Her findes højst et par hundrede tegn, hvorimod antallet af ord i en ordbaseret model ofte skal tælles i millioner, hvilket gør denne metode mere beregningsmæssigt effektiv. For at kunne benytte disse embeddings, trænes en separat character language model, på et stort, ikke-annoteret dataset. Denne type model kan udregne sandsynligheden for det næste tegn i en sætning, ud fra de forrige tegn, og dermed opfange den statistiske struktur i det korpus som modellen er trænet på. På figur 3.6 er vist hvordan disse character embeddings konstrueres.

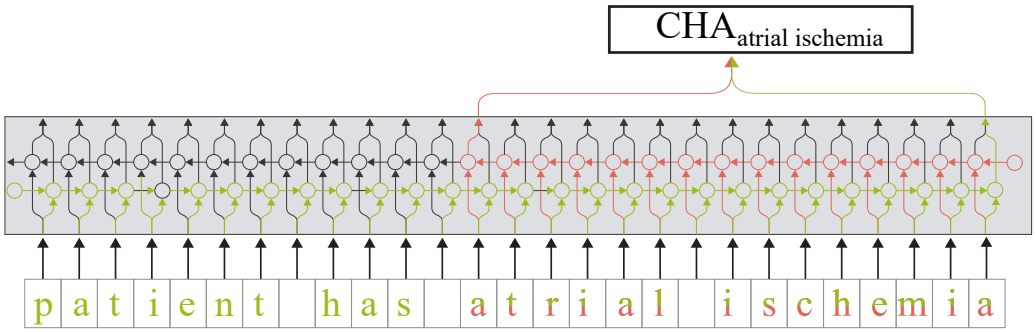


**Fig. 3.6.:** Figuren viser hvordan disse kontekstualiserede embeddings skabes ved hjælp af et RNN. I et character baseret RNN, tages det forrige input i betragtning. Den grønne del af modellen er forward modellen, og den røde model er backward modellen.

På figuren ses to modeller: forward modellen markeret med grønt, og backward modellen markeret med rødt. For forward modellen gælder det, at outputvektoren for den sidste karakter i det ord som skal embeddes, bliver sammensat med outputtet fra backward modellen, fra den første karakter i denne. Denne metode medfører at den konstruerede characterembedding modtager information om sætningen, både fra venstre mod højre, og omvendt.

Til at konstruere disse embeddings blev Flair frameworket benyttet [59]. Skaberne af frameworket beskriver ikke hvilke indstillinger der fungerer bedst, og derfor er standard indstillingerne benyttet. Der blev, som i Word2Vec modellen, ikke benyttet en prætrænet model. Hyperparameterne for både word og character embeddings er vist i tabel 3.1

| Statisk Embedding | | Dynamisk Embedding | |
|---|---|---|---|
| Embedding | Word2Vec | Embedding | Ch. emb* |
| Type | Negative sampling | Type | GRU |
| Dim. | 100 | Antal lag | 1 |
| Neg. samples | 20 | Sekvenslængde | 100 |
| Epochs | 10 | Dim | 128 |
| Learning rate | 0.025 | Learning rate | 0.01 |
| Korpus | EKG(MIMIC) | Korpus | EKG(MIMIC) |

**Tab. 3.1.:** Tabellen viser de benyttede hyperparametere for Word2Vec og character embeddings. I tabellen er de vigtigste hyperparameters illustreret, og derfor adskiller de valgte hyperparametere sig fra Word2Vec og character embeddings. *Det skal bemærkes at der benyttes to modeller til at skabe character embeddings; forward modellen(FW) og backward modellen(BW).

For begge typer af embeddings gælder det, at testsættet ikke er blevet benyttet til at konstruere de valgte embeddings. Dette er gjort for at undgå data-leakage, hvilket kan medføre et overoptimistik resultat, som vil performe godt på testsættet, men som ikke vil performe tilsvarende ved en implementering. Dette betyder at ingen af de dataeksempler der findes i testsættene, er benyttet til træning af embeddings.

### 3.3.3 Detektering af kandidatbegreber

*Dette afsnit er opdelt i to dele. Først beskrives active learning som en metode til effektiv annotering af data, der skal benyttes til de anvendte neurale netværk. Dernæst beskrives det neurale netværk, der er benyttet til at detektere kandidatbegreber.*

#### Active learning

Som i afsnit 2.2.3 blev det belyst, at manglen på annoteret data kan være et problem for NLP-algoritmer. I dette projekt er der benyttet active learning, for at håndtere denne udfordring. Denne tilgang har i flere studier vist sig at effektivisere annoteringsarbejdet [29, 50]. Indenfor det kliniske domæne har Kholghi m.fl. [30] vist, at tiden det tog at annotere på udskrivningsnotater, kunne reduceres med op til 35%. Ved at placere modellen i annoterings-loopet er det muligt at nedbringe omkostninger, og at reducere den tunge arbejdsopgave det kan være at annotere data. Der findes flere former for active learning, hvor en af de mest almindelige former er den såkaldte least confidence sampling[7]. Denne metode er blevet benyttet i dette projekt.

Metoden går ud på først at sortere tekstsekvenser, fra sandsynlig til mindre sandsynlig, og dernæst udtrække de eksempler algoritmen er mest i tvivl om. Alle tænkelige kombinationer af sekvenser kan selvfølgelig ikke evalueres, så det er udelukkende det prædikterede output med den højeste sikkerhed, der benyttes.

Alle teksteksempler normaliseres derefter, i forhold til den beregnede usikkerhed, således at summen for alle giver én. Derefter udtrækkes samples fra sandsynlighedsfordelingen. Dette medfører at træningsalgoritmen præsenteres for sikre eksempler, dog ikke så hyppigt som den præsenteres for usikre eksempler. Dette gøres for undgå unødig bias. Når en række eksempler er blevet annoteret, gentrænes modellen. Sandsynlighedsfordelingen laves ved at normalisere de prædikterede sekvenser, i forhold til summen af alle prædikterede sekvenser. Active learning metoden er illustreret på figur 3.7. I dette projekt benyttes Prodigy [19] frameworket, som er et Python/web-baseret framework, til annotering af tekst-dokumenterne. Dette framework er valgt pga. frameworkets simplicitet, og pga. dets hjælpefunktioner der kan benyttes til active learning.

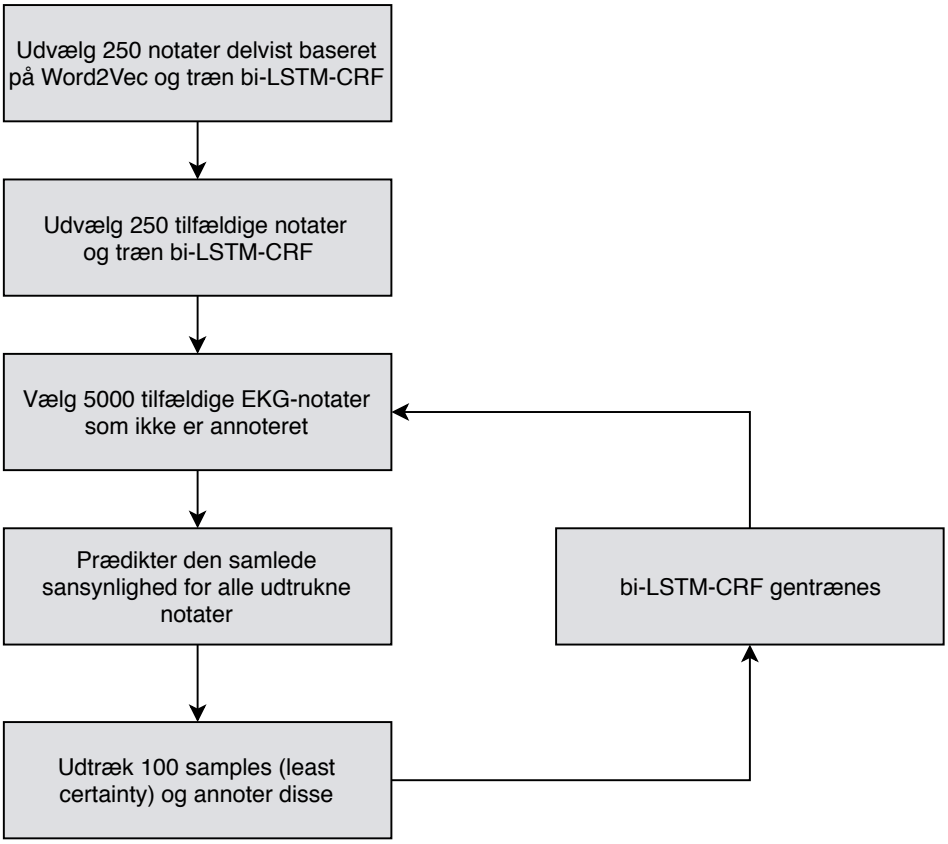


**Fig. 3.7.:** Figuren viser de forskellige steps for active learning strategien. Først udvælges 250 samples tilfældigt. Dernæst benyttes wordembeddings sammen med LSTM-modellen til at finde finde og annotere relaterede diagnoser. Herefter følger en iterativ proces, hvor de mest usikre eksempler udvælges og annoteres.

For at gøre brug af en active learning strategi, er det nødvendigt i starten at udvælge nogle samples tilfældigt, da modellen ikke er trænet på data. Derfor blev 250 tilfældigt udvalgte samples annoteret. Ud fra 250 samples blev de annoterede diagnoser noteret, og disse blev så benyttet til at finde variationer af de fundne entiteter, samt nye entiteter, ved hjælp af wordembeddings. Dette blev gjort for at indsamle flere forskellige diagnoser, der kunne skabe mere variation i datasættet. Et konkret eksempel som blev benyttet, var at først blev diagnosen "infarct"fundet. Ved at søge på de ordvektorer der havde en lille cosinus-distance fra “infarct”, kunne ord som “infarction”, “mi”, “bvh” og "extrasystoles” detekteres. Et ord som “bvh” vil ellers være vanskeligt at kunne detektere, da det en forkortelse af “biventricular hypertrophy”, hvilket er et begreb der kun optræder 250 gange i de mere end 200.000 EKG-dokumenter (forkortelsen bvh optræder seks gange). Med denne metode blev der opsamlet yderligere 250 samples. Figur 3.8 illustrerer hvordan disse ordvektorer kan benyttes til at finde semantisk lignende begreber.

Et visuelt eksempel på fremgangsmåden er illustreret på figur 3.8. I figuren er ordene projekteret ned i 2D ved hjælp af t-SNE [20]. Ud af ca. 15.000 forskellige ord er disse

de nærmeste naboer til ordet “infarction". Ved word embeddings vil relaterede ord ofte ligge tæt op ad hinanden, som i det illustrerede eksempel. Her ses et eksempel på at ord der befinder sig i nærheden af "infarct” er “mi” (myocardial infarct), og øvrige stavemåder af infarctions, som f.eks. “Infacation” og “infarctions”.

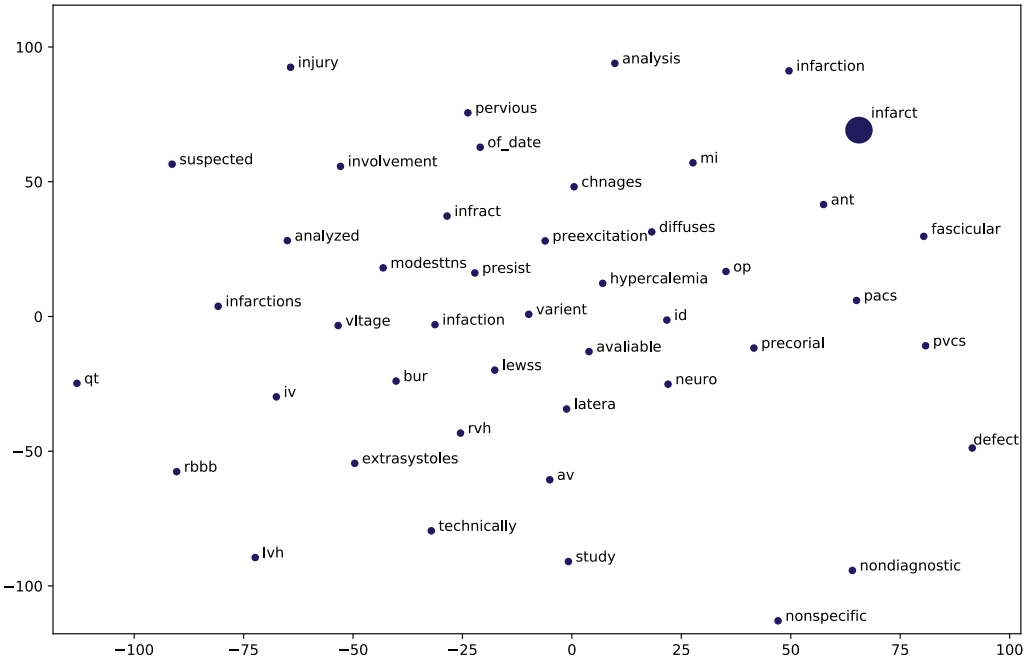


**Fig. 3.8.:** Figuren viser en 2D projektion af Word2Vec vektorene, der er blevet trænet på EKG-noterne i MIMIC datasættet. Der er for mange ord i datasættet til at alle ord kan visualiseres, så i det illustrerede eksempel er vist de tætteste naboer til ordet “infarct”.

Active learning strategien blev benyttet indtil loss-værdien stabiliserede sig. Loss værdien er mål for hvor godt modellen performer på dataeksemplerne, og hvis denne værdi ikke faldt indenfor 100 eksempler blev samplingen afbrudt. Det genererede datasæt var som følgende. :

| Antal diagnoser | Positiv | Negering | Ubekræftet |
|---|---|---|---|
| Træning/validering | 1725 | 185 | 608 |
| Testsæt | 273 | 29 | 106 |

**Tab. 3.2.:** I tabellen ses antallet af annoterede diagnoser.

### LSTM

I dette afsnit beskrives modellen benyttet til Named-Entity-Recognition (NER). Neurale netværk kommer i forskellige udgaver, hvoraf de mest benyttede til NER er recurrent neural networks og convolutional neural networks. Da entity recognition er et problem hvor der indgår sekvensafhængigheder, kan dette modelleres af et recurrent neural network. Et recurrent neural network har et såkaldt feedback-loop i dens arkitektur, der muliggør udnyttelse af den tidsafhængighed der er i sekvenser, hvilket kvalificerer denne type arkitektur til at blive benyttet til NER. Et neuralt netværk trænes ved hjælp af back propegation, som er en metode til at justere parametrene i modellen. Dette gøres ved at beregne gradienterne for parametrene, for derefter at justere modellens parametre ved hjælp af disse. Et tradionelt RNN har dog den ulempe, at denne type arkitektur kan lide under at gradienterne bliver for små, eller alt for store, under træning af modellen. Dette medfører at denne type kan have svært ved at lære længere afhængigheder. Derfor benyttes oftest RNN, som benytter særlige "units" i stedet for dem, der er at finde i traditionelle RNN. Ofte benyttede arkitekturer er long-short-term memory (LSTM) netværker, eller gated recurrent netværker (GRU) i stedet. Disse har flere parametre end traditionelle netværk, og studier har endnu ikke kunne identificere, om GRU eller LSTM arkitekturen er bedst. Den foreløbige konklusion er, at det vil afhænge af det specifikke problem [27]. Dog fandt Chung m.fl. [8], at begge arkitekturer har en bedre performance end et traditionelt RNN [8]. En uddybende forklaring af LSTM kan findes i appendix B.

I dette projekt benyttes en bi-directional LSTM-CRF arkitektur beskrevet af Yu m.fl. [63]. Denne metode har vist sig at være state of the art på flere NER benchmark-datasæt, hvis denne kombineres med character embeddings, sammenkoblet med statiske vektorer [59]. Arkitekturen er vist på figur 3.9

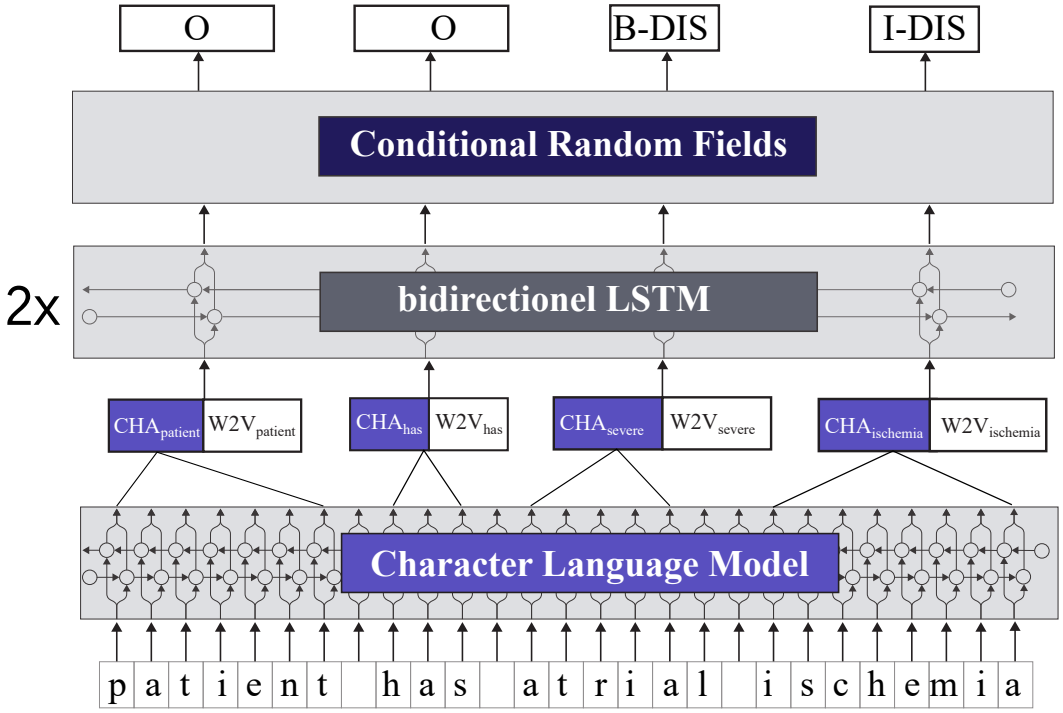


**Fig. 3.9.:** Figuren viser arkitekturen, af den benyttede NER-model. I bunden af figuren kan det ses hvordan character embeddings kontrueres, og hvordan disse sættes sammen med Word2Vec embeddings.

Først genereres embeddings som beskrevet i afsnit 3.3.2. Disse føres ind i bidirectional LSTM-lag, hvilket medfører at modellen kan benytte input effektivt, fra tidligere og fremtidige input. Dette betyder at inputsekvensen sendes igennem både fra start til slut og fra slut til start. Outputtet fra dette lag føres videre ind i et tilsvarende LSTM-lag, og outputtet herfra føres så ind i et CRF lag. Ved hjælp af CRF-laget udvælges den sekvens af outputs, der samlet set har den højeste sandsynlighed. Dette medfører at CRF-lagets opgave tilføjer begrænsninger til den prædikterede sekvens. Disse begrænsninger kan læres under træningsprocessen, og optimeres vha. gradient descent. En sådan begrænsning kan være at modellen ikke må prædiktere et I-tag (inside) før et B-tag (begin). [32] Outputtet fra CRF-laget, er sandsynligheden for at en given token tilhører et bestemt tag.

**Træning af NER-modellen**

Efter at arkitekturen var valgt, blev automatisk hyperparametersøgning anvendt, og til dette er random search benyttet. [1]. Ved denne metode specificeres en række hyperparametere, hvor der udvælges et sæt af tilfældige hyperparametere, ud fra de udvalgte hyperparametere. Antallet af søgeiteratoner blev sat til 50.

| Hyperparametre | | Benyttet |
|---|---|---|
| Antal LSTM lag | [1,2,3] | 2 |
| Dropout | [0.0-0.8] | 0.2 |
| Batch size | [8,16,32,64] | 16 |
| Hidden size | [64,128,256,512] | 128 |
| Learning rate | [0.1,0.01,0.001] | 0.1 |
| RNN type | [GRU, LSTM] | LSTM |

**Tab. 3.3.:** Tabellen viser de fundne hyperparametre der blev benyttet til NER-modellen.

Antallet af lag blev sat til et minimum på et og et maksimum på tre. Oftest benyttes få lag i LSTM netværk. I studiet Vollgraf m.fl. [59] blev benyttet ét LSTM lag, for at opnå state-of-the art resultater. Det formodes derfor ikke at flere end tre lag vil være fordelagtigt. Til at regulere modellen, og dermed undgå overfitting, blev dropout benyttet. Dropout medvirker at dele af netværket midlertidigt fjernes under træning. Dette tvinger netværket til at dens neuroner ikke udelukkende lærer bestemte features. Efter træning af modellen, deaktiveres dropout. Her blev søgt fra 0 til 80% dropout. Batch size angiver hvor mange træningseksempler der indføres i netværk på træningsiteration. En lavere batchsize har ofte en regulerende effekt, og til søgning over batch size blev 8-64 træningseksmpler benyttet. Hidden size angiver dimensionaliteten på “hidden state” vektorerne i netværket. En for lille størrelse, vil resultere i at netværket ikke opfanger den statistiske struktur tilstrækkeligt, og et for stor antal kan medføre overfitting. Learning rate angiver hvor hurtigt netværket lærer ved hjælp af gradient descent. En for lille learning rate vil resultere i en længere træningstid, hvor en for stor learning rate vil medføre at netværket ikke konvergerer. Til træning af modellen blev benyttet et område fra 0.1 til 0.001. Der blev ikke forsøgt en højere learning rate, da indledende forsøg viste at modellen ikke performede godt ved en højere værdi.

Der blev også søgt over to forskellige RNN arkitekturer. Der blev benyttet de to mest almindelige former for RNN. Forskellen mellem de to er, overordnet set, at LSTM har flere parametre der skal trænes i forhold til GRU. Typisk for NER er, at tekst opsplittes i sætninger, hvor disse benyttes som input. Det blev forsøgt at benytte sætninger, men dette blev fravalgt grundet en længere træningstid, samt intet øget udbytte i forhold til performance. Helheden af et givet notat blev i stedet anvendt som input. Derudover benyttes "IOB"annoteringsmetoden, som vist i afsnit 2.1. Dette betyder at den første token i en diagnose har et andet "tag"end tokens der ikke er den første i en diagnose. Grunden til dette er at det skal være muligt at kunne adskille to eller flere diagnoser, der lægger sig op ad hinanden uden mellemliggende tokens.

### 3.3.4 Detektering af kontekst for kandidatbegreber

Når en diagnose er identificeret, skal denne klassificeres i forhold til hvilken kontekst denne har. Dette er gjort ved at benytte en textclassifier. En textclassifier tager et stykke tekst som input, og outputtet vil angive hvilke klasser denne tilhører. Da en sætning kan indeholde flere diagnoser, er det nødvendigt at kunne markere hvilken diagnoser man ønsker konteksten for. Metoden til at gøre dette er inspireret af Chen [6], hvor såkaldte position markers blev benyttet til at indikere det ord konteksten skulle prædikteres for. Et eksempel på dette vil f.eks. være sætningen "The patient denies chest pain." vil blive ændret til sætningen: "The patient denies <c> chest pain </c>.", hvor <c> og </c> indikerer starten og slutningen på den diagnose hvor konteksten skal klassificeres.

Til at klassificere konteksten blev frameworket Universal Language Model Finetuning (ULMFiT), som består af et LSTM-netværk, benyttet. ULMFiT har vist sig at have en høj performance på en række forskellige datasæt indenfor en række domæner [23]. Modellen udmærker sig derudover ved at have vist at kunne håndtere datasæt med relativt færre annoterede eksempler end tilsvarende tekstklasssifikationsmodeller. Modellen er en semisuperviseret model, hvilket vil sige at modellen er trænet på en kombination af annoteret og ikke-annoteret tekst. Denne metode udmærker sig ved at kunne forøge performance uden annoterede træningseksempler, ved at lære strukturen af data fra det ikke-annoterede data [42]. Dette medfører en potentiel reducering af tid og omkostninger, der ellers kan være forbundet med at konstruere modellen. Til at træne modellen med ikke-annoteret data, benyttes flere notater fra MIMIC databasen, hvor størstedelen var andre typer notater end EKG-notater. Da der findes over en million notater i MIMIC databasen, blev en delmængde af de tilgængelige notater benyttet.

Modellen der benyttes til at lære fra de ikke-annoterede eksempler, er den såkaldte AWS-LSTM model, [38] som er et LSTM-netværk med tre lag. Denne er en language model, hvilket vil sige at modellen kan prædiktere det næste ord, i en sekvens af ord. Det er efter samme princip som modellen der blev benyttet til at generere character embeddings, beskrevet i afsnit 3.3.2, bare hvor hele ord benyttes som input, i stedet for karakterer. Overordnet set fungerer metoden ved at træne denne language model på det ikke-annoterede korpus, for efterfølgende at ændre modelarkitekturen til en classifier, hvor de annoterede eksempler benyttes til træne modellen færdig. De forskellige dele af modellen er illustreret på figur 3.10. Det er vigtig at pointere at det er de samme LSTM lag, med de samme parametre, der benyttes i de tre dele. De tre dele er angivet som; Generel-domæne prætræning ved hjælp af en language model (GPLM), target task language model fine-tuning (TTLMFT), og target task classifier fine-tuning (TTCF), og disse er yderligere uddybet i dette afsnit.

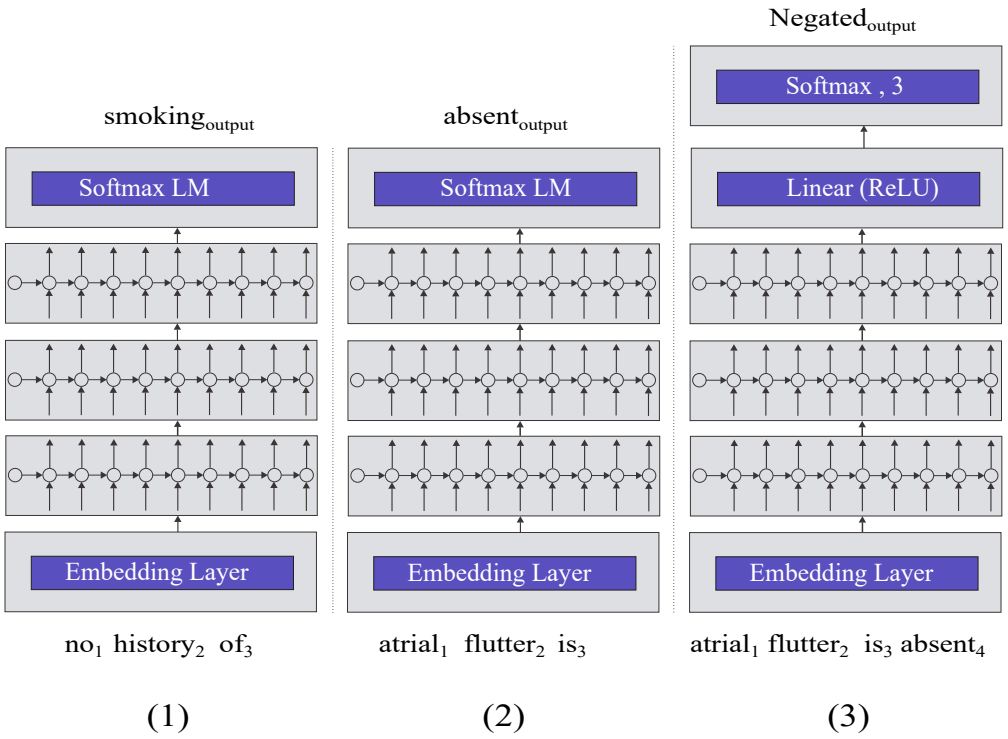


**Fig. 3.10.:** Figuren viser arkitekturen af den benyttede model. I 1 og 2 er language modellen illustreret, og i 3 er classifieren vist. 1 viser language modellen, hvor den trænes med tekst fra MIMIC datasættet, og i dette step benyttes ikke EKG-notater. Her ses at "no history of"er input til modellen, hvor "smoking"er output. I 2 trænes videre på ikke-annoterede EKG-notater, ved samme metode som i 1, og her ses at "atrial flutter is"er input, og "absent"er output. I 3 udskiftes det sidste softmax-lag fra language modellen, til et lineært lag efterfulgt af ReLu aktiveringsfunktioner, samt et nyt softmax lag. I dette step benyttes annoteret tekst. Her benyttes sætningen "atrial flutter is absent"som input og outputtet er en af de tre kontekstklasser.

**Generel-domæne prætræning ved hjælp af en language model (GPLM).**
Alle noter blev præprocesseret efter samme fremgangsmåde, som beskrevet i afsnit 3.2. I denne fase er det meningen at modellen skal opfange generelle egenskaber om medicinsk tekst.

Der trænes på i alt 502.365 notater. Henholdsvis fra tabellerne:

- Respiratory - 31.739 notater
- Physician - 141.624 notater
- Echo - 45.794 notater
- Nursing - 23.556 notater

- Discharge - 59.652 notater

**Target task language model fine-tuning (TTLMFT)**
Derefter trænes der videre med ikke-annoteret domænespecifik data. Dette gøres fordi det kan antages at den domænespecifikke data har en distribuering, der adskiller sig fra dataen fra det generelle domæne. I denne fase benyttes 200.051 EKG-notater.

**Target task classifier fine-tuning (TTCF)**
Til sidst fintunes modellen gradvist med labelled data. I denne fase benyttes diskriminativ fintuning, hvilket betyder at der benyttes en lavere learning rate i de første lag af netværket. Til finetuning af classiffieren tilføres to lineære lag med ReLu aktiveringsfunktioner, batch-normalization og dropout. Det sidste lag i classifieren er et softmax-lag hvor outputtet er en sandsynlighedsdistrubition over de tre klasser. I denne fase blev 2048 annoterede diagnoser benyttet.

**Træning af klassifikationsmodellen**
Eftersom modellen tager betydeligt længere tid at træne end de andre modeller, benyttes automatisk søgning efter hyperparametre ikke, da dette kræver megen repetition af modellernes træning, hvilket vil være for tidskonsumerende. Standard parametre for regulering benyttes, samt ADAM optimizer til at opdatere modellens parametre. Denne optimizer har empirisk vist sig af virke godt med UMLFiT [23]. Learning rate findes vha. tilgang beskrevet af Smith [53]. Ved denne metode startes med en lav learning rate, hvorefter denne forøges eksponentielt for hvert batch der trænes på. Ved at plotte loss værdien for hvert batch i én epoch, i forhold til learning rate, kan den optimale learning rate findes. Dette gøres ved at vælge en learning rate startende i området omkring det stejleste fald, inden området hvor loss-værdien stiger eksponentielt. Denne er illustreret på figur 3.11.

De forskellige hyperparametre er beskrevet i tabel 3.4

| | Learning rate | Dropout | Batchsize | Epochs |
|---|---|---|---|---|
| GPLM | 0.005 | 0.4 | 32 | 10 |
| TTLMFT | 0.1-0.001 | 0.4 | 32 | 12 |
| TTCF | 0.1-0.001 | 0.5 | 32 | 5 |

**Tab. 3.4.:** De valgte hyperparametre for UMLfit

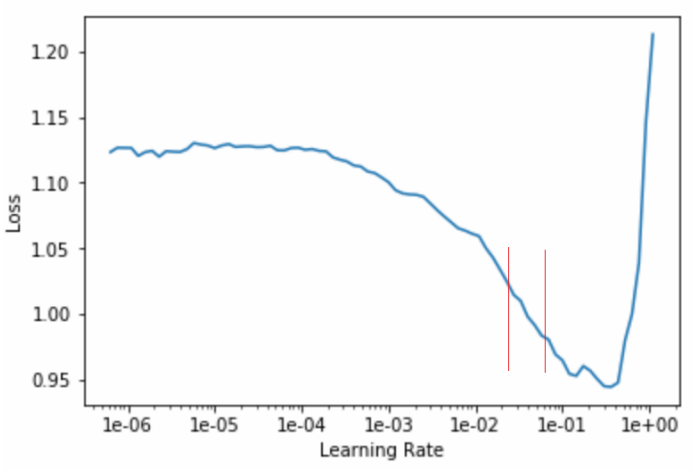


**Fig. 3.11.:** Figuren illustrerer metoden til at finde den optimale learning rate for modellen. Den røde streg viser den benyttede learning rate, hvilket er 0.005.

### 3.3.5 Mapning til SNOMED CT

I dette afsnit beskrives fremgangmetoden med at mappe kandidatbegreber til SNOMED CT koder. Der blev gjort brug af en Python-client der kunne tilgå en rest-API fra UMLS [49]. Ved at sende en forespørgsel med et kandidatbegreb kunne der returneres en unik SNOMED CT kode. Det blev dog fundet at forespørgslen skulle være meget præcis for at kunne returnere en kode. Det vil sige at f.eks. forkortelser, tegn, synonymer eller bøjninger kunne medføre at et kandidatbegreb ikke kunne mappes til et SNOMED begreb. For at imødekomme denne udfording blev en såkaldt "sieve based"metode benyttet, hvilket er inspireret af en metode af D'Souza og Ng [14]. Metoden går ud på at modificere kandidatbegrebet i flere trin, og i en bestemt rækkefølge. Efter hvert trin, søges der på det modificere kandidat begreb. I algoritmen af D'Souza og Ng [14] benyttes en liste med synonymer til at modificere kandidatbegreberne. I dette projekt benyttes i stedet word vectors, beskrevet i 3.3.2, til at finde beslægtede termer. Algoritmen var som følgende: 1) Først benyttes den eksakte string til forespørgslen. 2) Fjernelse af bindestreg. 3) I dette step benyttes en liste med forkortelser fra [18] til at ændre forkortelser til den fulde form af et begreb. 4) I dette step ændres cifre fra 1-10 til en ordform. Dette betyder f.eks. at 1 bliver til one. Derudover ændres 1st til first, osv. Til håndtering af synonymer i dette step benyttes word vectors til finde relaterede termer. Dette gøres ved at sammenligne cosinus distancen, mellem ordvektoren for det detekterede begreb og alle begreber der blev opsamlet under annotering.

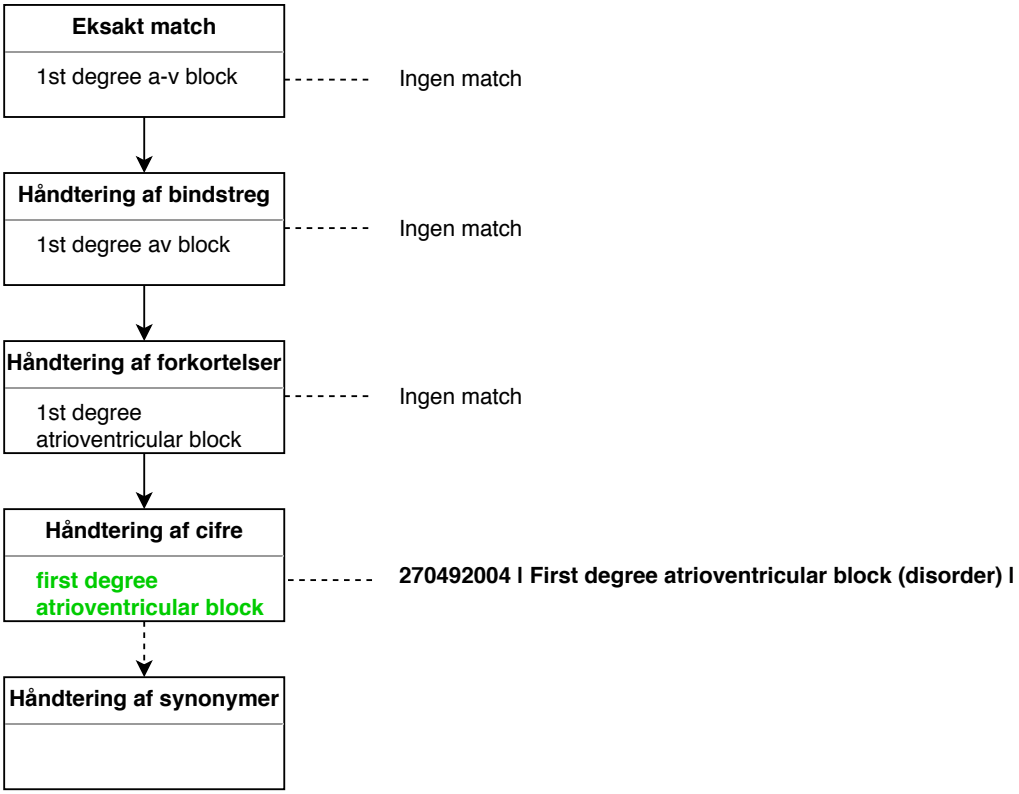


**Fig. 3.12.:** Figuren illustrerer, hvordan en forespørgsel laves til UMLS rest-API for hvert step.

## 3.4 Evaluering

I dette afsnit vil de benyttede evalueringsmetoder blive beskrevet for NER-algoritmen, kontekst classifieren og systemet benyttet på CLEF datasættet. De benyttede metoder til evaluering er udvalgt eftersom disse er typisk anvendte metoder til evaluering indenfor named entity recognition, tekstklasssifikation og klinisk entity recognition. Ydermere vurderes det at metoderne giver et transparent billede af modellens performance.

Derudover blev modellernes robusthed testet, ved at udføre en analyse af enkelte eksempler. Dette blev analyseret, for at give et indblik i hvordan algoritmen optræder og hvorfor modellerne agerer som de gør. Dette giver en kvalitativ indsigt i hvordan algoritmen opererer, og denne viden kan danne basis for forståelse i forhold til hvilken retning en eventuel videreudvikling af systemet ville skulle gå. Alle rapporterede resulter vil stamme fra de respektive testsæt.

### 3.4.1 Detektering af kandidatbegreber

Til evaluering af denne del af projektet benyttes precission, recall og F1-score som den primære evalueringsmetrik.

Precision og recall er givet ved følgende formler:

$$precision = \frac{TP}{TP + FP} \tag{3.1}$$

$$recall = \frac{TP}{TP + FN} \tag{3.2}$$

F1-scoren er givet ved:

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{3.3}$$

I formlerne er TP sand positive, FP angiver falsk positive og FN angiver falsk negative. Precision fortæller om hvor stor en andel af de detekterede diagnoser der egentlig var diagnoser, og recall er et mål for hvor mange rigtige diagnoser der blev fundet. For evalueringen gælder det, at beregningen er baseret på alle tokens der er tilknyttet en klasse, og ikke tokens der ikke er tilknyttet en klasse, da dette vil resultere i et overoptismisk resultat. Precission og recall er beregnet ud fra kun eksakte match på en diagnoser tælles som korrekt. Dette betyder, at hvis kun dele af diagnosen detekteres, så tæller det ikke som værende korrekt.

#### Uventede prædiktioner

Fire eksempler fra testsættet blev undersøgt, hvor algoritmen opførte sig anderledes end forventet. Disse fire eksempler blev udvalgt på baggrund af, at de beskrev forskellige aspekter af uventede prædiktioner. Her blev der både givet eksempler på hvornår algoritmen fungerede dårligere end forventet, men også hvornår den prædikterede bedre end forventet.

#### Robusthedsanalyse

Her blev en sætning permuteret, for at undersøge grænserne for hvornår den kunne prædiktere et ord. Dette blev gjort ved at indsætte stavefejl, samt indsætte forkortelser.

### 3.4.2 Detektering af kontekst for kandidatbegreber

Det blev evalueret, hvor godt algoritmen kunne detektere konteksten som et givet ord befandt sig i. Denne score er også angivet som F1-score, på samme vis som ved evalueringen af NER-algoritmen.

#### Evaluering af enkelte eksempler

I denne undersøgelse blev få eksempler brugt, for at undersøge hvorfor kontekst classifieren agerede som den gjorde. Dette blev gjort for at undersøge hvilke ord der var afgørende, for at en sætning blev klassificeret som den gjorde. For at undersøge dette, blev gradienten af ouputsandsynligheden i forhold til inputtet beregnet og visualiseret. Dette blev beregnet vha. Python bibliotektet FastAI [24].

### 3.4.3 Mapning til SNOMED CT

Afslutningsvist blev systemet testet på ShARe datasættet, hvor både diagnoser samt tilhørende Snomed CT kode skulle detekteres. Her vil både NER-algoritmen, samt mapningalgoritmen komme i spil. Til evaluering af dette blev F-score opdelt i strict F og relaxed F-score. Ved strict F1 gælder det, at en diagnose er sand positiv hvis det detekterede span er præcis det samme som angivet i testsættet. Derudover er det krav, at den detekterede kode er identisk med CUI'en i testsættet. Ved relaxed F-score gælder ligeledes, at CUI'en skal være identisk med kode i test, dog er der ikke krav om at span'et skal matche eksakt.

# 4 Resultater

I dette kapitel præsenteres resultaterne for henholdsvis NER, kontekstklassifikation og mapning til SNOMED CT begreber. For hver af de enkelte dele vil både den overordnede evaluering, og evaluering af enkelte testeksempler blive beskrevet i de respektive afsnit.

### 4.0.1 Detektering af kandidatbegreber

I denne del benyttes trænings- og testsættet for NER-modellen. Der var 1606 noter, hvor en andel på 15% af noterne udgjorde testsættet, og en andel på 15% udgjorde valideringsættet. Resultatet er rapporteret på tabel 4.1 og et mere detaljeret billede af algoritmens prædiktioner er vist på figur 4.1.

| Precision | Precision | Recall | F1 |
|---|---|---|---|
| Valideringssæt | 0.89 | 0.91 | 0.90 |
| Testsæt | 0.88 | 0.91 | 0.89 |

**Tab. 4.1.:** Tabellen viser precision og recall for NER-algortimen, for henholdsvis valideringsog testsættet.

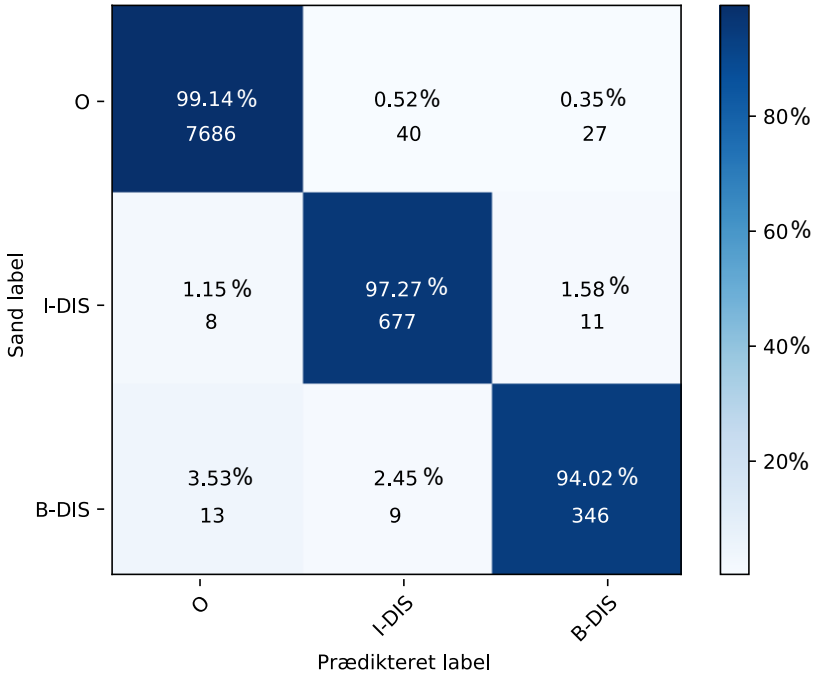


**Fig. 4.1.:** Figuren viser en confusion matrix for diagnose detektering. Der er angivet både den procentvise andel af prædiktioner, samt det absolutte antal. "O"tagget står for "outside", hvilket betyder at en token ikke er en diagnose. "I"(inside) dvs. en token der ikke er den første i en diagnose. "B"(begin) tagget er den første token i en diagnose.

Data i confusion matrix blev angivet i procent, samt i absolutte tal, grundet en naturlig overvægt af “O” tags. Relativt set var det også dette tag, som havde den højeste accuracy. Der hvor “O” tagget hyppigst blev prædikteret forkert, var ved “I-POS” tagget. Dette kunne tyde på at algoritmen har sværest ved at vide hvornår et term slutter, i forhold til at vide hvornår et term starter, hvis det rigtige tag er “O”. Det bør også bemærkes at den største procentvise fejl, var når det rigtige label var “B-DIS”, men den prædikterede i virkeligheden var et “O” tag, hvilket tyder på algoritmen havde mest besvær med at finde ud af hvornår termer startede. I afsnit 4.0.2 gennemgås få eksempler, for at kunne give en kvalitativ indsigt i hvorfor det forholdt sig sådan.

### 4.0.2 Eksempler og forklaringer på korrekte og inkorrekte udfald

I dette afsnit gennemgås enkelte eksempler. I de følgende eksempler er prædiktioner, hvor algoritmens output er tilsvarede de annoterede tokens, markeret med blå. Annoterede tokens, som ikke blev detekterede er markeret med grøn. Hvis algoritmen detekterede en token som ikke var annoteret, er denne markeret med gul. Teksterne

vil, i nogle eksempler, være forkortet. De første to eksempler viser hvor algoritmen laver fejldetekteringer, og eksempel 3-4 viser hvor algoritmen klarer sig bedre end forventet.

I eksempel 1 prædikterer algoritmen “injury", hvor det korrekte er “myocardial injury”. “myocardial injury” var ikke at finde i træningssættet, men fandtes i testsættet, og dette forklarer højst sandsynligt hvorfor denne diagnose ikke blev detekteret. Derimod fandtes både ordet “myocardial” og “injury” hver for sig i træningssættet. I dette tilfælde var algoritmen 63% sikker på at “myocardial” ville have et “O” tag.

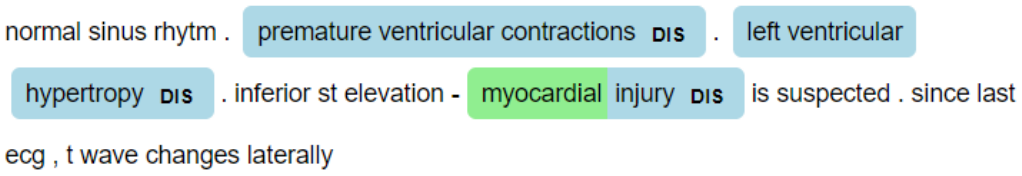


**Fig. 4.2.:** Eksempel 1. Teksten viser at algoritmen ikke opfangede en diagnose korrekt.

I eksempel 2 vises et eksempel hvor to simultane fejl medfører en fejldetektering. Her var "a-v coonduction"annoteret som target, men tokenization fejl, samtidig med en stavefejl i ordet "co**o**nduction", gjorde at algoritmen prædikterede "A-V". Hvis kun en af fejlene havde været til stede ville algoritmen prædiktere det rigtige term. Normaltvis ved tokenization på engelsk, vil “-” være en token for sig selv. Der blev lavet manuelle undtagelser for dette i dette projekt, bl.a. Ved s-t, q-r og r-interval. Dette blev ikke gjort for “a-v”, som blev til “a - v”. Dette betyder at i stedet for én token, så blev dette til tre tokens, og dette har sandsynligvis medført denne fejl.

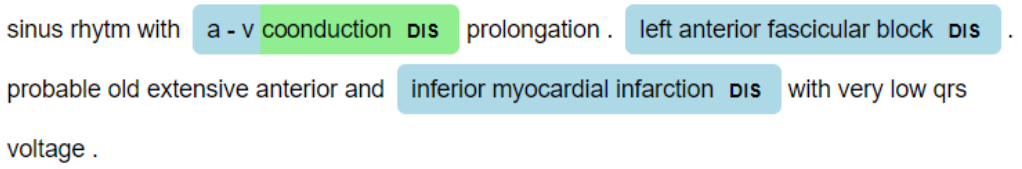


**Fig. 4.3.:** Eksempel 2. To mindre simultane fejl medfører en fejldetektering.

I eksempel 3 detekteres ordet “atrial arrhythmias ” på trods af at ordet “arrhythmias ” ikke er at finde i træningssættet. Ordet “arrhythmia ” er derimod annoteret (uden s i slutningen). Eftersom der benyttes character embeddings, kan det nemmere lade sig gøre at finde ukendte variationer af ord, der ligner ord som algoritmen er blevet præsenteret for før.

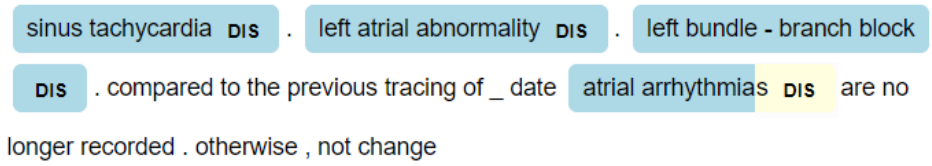


Fig. 4.4.: Eksempel 3. Eksempel på effektiviteten af characterembeddings.

I eksempel 4 er ordet “intraventricular conduction disturbance ” ikke blevet annoteret i datasættet, men denne blev alligevel detekteret. Dette kan skyldes at termer som “Non-specific intraventricular conduction delay” og “Intraventricular conduction defect” findes, og dette er et eksempel på at algoritmen har lært at generalisere.

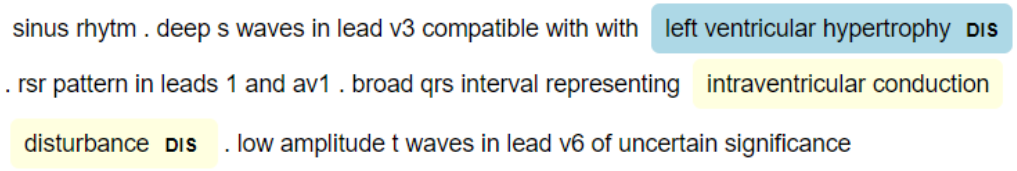


Fig. 4.5.: Eksempel 4. Et eksempel på at algoritmen kan prædiktere nye, usete ord.

### 4.0.3 Robusthedsanalyse

Figur 4.6 viser hvor meget en sætning kan ændres, før algoritmen prædikterer forkert. Linje nummer et viser baseline sætningen. Altså en sætning hvor de to diagnoser staves og prædikteres korrekt, og disse er markeret med grønt. I linje 2 flyttes rundt på to bogstaver i de to diagnoser, hvilket ikke betyder noget for prædiktionen. I linje 3 forekommer der to ændringer i “atrial fibrillation” som medfører at algoritmen ikke kunne detektere diagnosen. Havde der kun været en af fejlene tilstede ville algoritmen have kunnet detekteret det korrekt. I linje 4 ændres ordet "abnormality"til et synonym, som ikke var i træningssættet. Dette kan algoritmen stadig detektere, og en plausibel forklaring er at “anomalous” er blevet kodet ind i de benyttede word embeddings. I linje 5 er der indsat forkortelser. Her ses at atr. ikke er en brugbar forkortelse for atrial i ordet atrial fibrilliation, men at “bi. atr” er en brugbar forkortelse. Det kan tilføjes at udskiftes “atr.” med “a.” i “ atrial fibrillation”, så vil dette være en forkortelse som algoritmen vil kunne detektere.

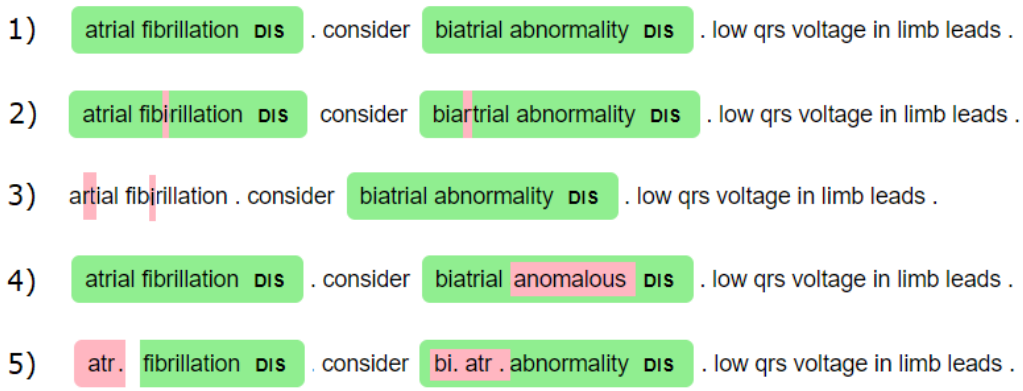


**Fig. 4.6.:** Figuren viser den sætning 1 der permuteres for at undersøge hvor meget der kan ændres i en sætning for at algoritmen stadig kan detektere de korrekte diagnoser.

## 4.1 Kontekstdetektion

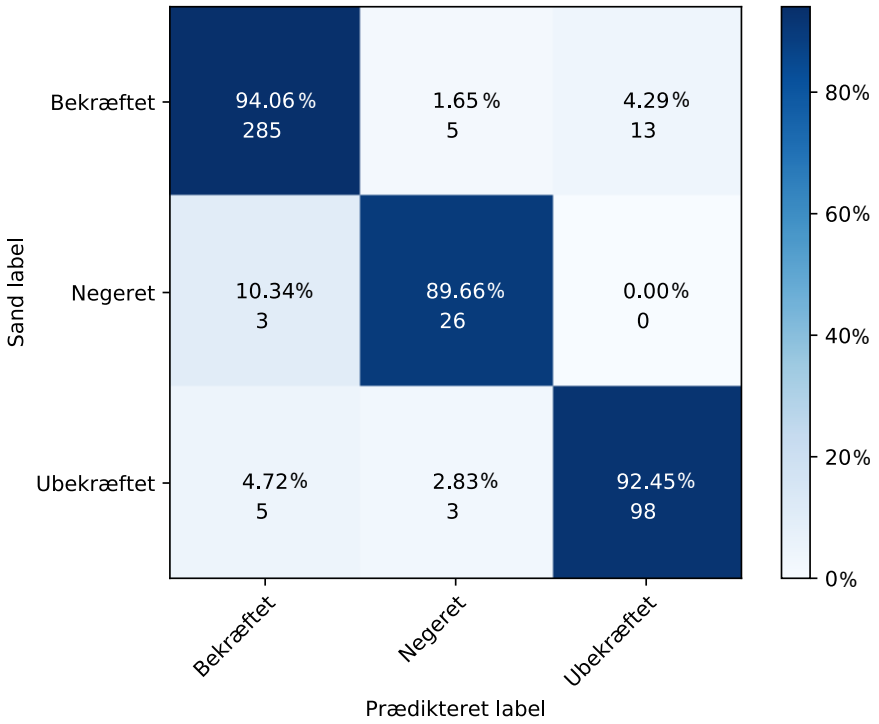


**Fig. 4.7.:** I figuren ses en confusion matrix for kontekst klassifikation. I figuren er både den procentvise og det absolutte antal for prædiktioner vist.

I figur 4.7 kan det ses at algoritmen var bedst til at detektere positive diagnoser. Der hvor algoritmen havde størst tendens til at prædiktere forkert, var ved negative diagnoser.

| | Precision | Recall | F1-score | Antal |
|---|---|---|---|---|
| Bekræftet | 0.97 (0.96) | 0.94 (0.97) | 0.95 (0.96) | 303 (309) |
| Negeret | 0.76 (0.89) | 0.90 (0.78) | 0.92 (0.83) | 29 (32) |
| Ubekræftet | 0.88 (0.93) | 0.92 (0.92) | 0.88 (0.92) | 106 (97) |
| | | | | |
| Micro | 0.93 (0.95) | 0.93 (0.95) | 0.93 (0.95) | 435 (438) |
| Macro | 0.92 (0.93) | 0.91 (0.89) | 0.92 (0.91) | 435 (438) |

**Tab. 4.2.:** I tabellen er vist de rapporterede klassifikationsmetrikker. Tallene i parenteser er validationssættet.

Det ses i tabellen, at især ubekræftede diagnoser havde en højere recall, relativt set. Dette betyder at evnen til at finde alle ubekræftede diagnoser er bedre, end dens evne til at detektere dem præcist. Sagt på en anden måde, den prædikterer for mange ubekræftede diagnoser i forhold til hvor mange der er.

### 4.1.1 Kontekstklassificeringseksempler

Jo mørkere det markerede tekst er, jo vigtigere er ordet for prædiktionen. Specielle tokens, som markerer starten og slutning på et kontekst ord, er filteret ud. Ligeledes er specielle tokens som markerer starten og slutningen på en sætning også blevet fjernet.

#### Tvivlsomme prædiktioner

I eksempel 1 er vist et eksempel hvor algoritmen ikke prædikterer korrekt(i forhold til label). Dette eksempel blev annoteret som en positiv diagnose. Algoritmen lægger vægt på "was not previously", men grundet ordene der kommer umiddelbart inden, skifter konteksten, da det er en sammeligning i forhold til sidste måling. Dette kan ses i eksemplet hvor ordet "previous " før kontekstordet, ikke detekteres som vigtig for outputtet, hvilket det burde have været.

```
rare atrial premature beat . non - specific low amplitude t waves
in leads i , avl and v4-v6 . int : possible inferior myocardial
infarction - old . a repeat electrocardiogram with a clear baseline would
be helpful . non - specific t wave abnormalities . consider clinical
correlation . compared to the previous tracing of _ date inferior
myocardial infarction was not previously present . t wave abnormalities
have appeared . tracing 1
```

**Fig. 4.8.:** Eksempel 1.Sand: Bekræftet, Prædiktion: ubekræftet.

Her ses et eksempel hvor ordet "atrial fibrillation" er annoteret som bekræftet, men hvor prædiktionen er ubekræftet. Scoren for bekræftet er relativt høj i forhold til de fleste prædiktioner, hvor der prædikteres ubekræftet. I dette tilfælde kan der muligvis argumenteres for at prædiktionen kunne tilfalde begge grupper.

```
irregular narrow complex tachycardia most consistent with atrial
fibrillation . rightward axis . possible anteroseptal myocardial
infarction of indeterminate age . diffuse non - specific st-t wave
changes . no previous tracing available for comparison .
```

**Fig. 4.9.:** Eksempel 2. Sand: bekræftet, Prædiktion: ubekræftet

### Svære detekteringer for algoritmen

I eksempel 3 blev "atrial fibrilation" detekteret til bekræftet, men det viste sig at den prædikterede negeret til 40% sandsynlighed. Ved flere lignende træningseksempler kunne det muligvis forventes at algoritmen vil kunne håndtere eksempler som disse. Overordnet set, var negerede den sværeste gruppe at detektere, og det var typisk eksempler som disse algoritmen havde svært ved.

```
probable atrial flutter with slow atrial rate and variable ventricular
response . compared to the previous tracing of  _date atrial fibrillation
has changed to atrial flutter .
```

**Fig. 4.10.:** Eksempel 3. Sand: negering Prædikteret: bekræftet.

### Algoritmens styrker

Eksempel 4 viser at algoritmen kan detektere den rigtige kontekst, hvis få ord udskiftes. Dette er med til at vise at algoritmen prædikterer på baggrund af de ord, der medfører at et term skifter kontekst. De forskellige kontekster blev detekteret korrekt, med sandsynlighed på over 80% for de tre forskellige eksempler. Den øverste sætning er fra testsættet, og i de efterfølgende er få ord blevet udskiftet for at ændre konteksten. For den ubekræftede diagnoser, og den negerede diagnose, kan det ses at ordene "suggesting" og "is not" er de vigtigste for disse prædiktioner. Overraskende viser eksemplet at "present" ikke er vigtigere end "shows", i eksemplet med den bekræftede diagnose.

Original(ubekræftet)

```
... st segment elevation in leads vi - v2 suggesting that anteroseptal
myocardial infarction is present of undetermined age...
```

Ændret(Bekræftet)

```
...st segment elevation in leads vi - v2 shows that anteroseptal
myocardial infarction is present of undetermined age...
```

Ændret(Negeret)

```
...st segment elevation in leads vi - v2 shows that anteroseptal
myocardial infarction is not present of undetermined age...
```

**Fig. 4.11.:** Eksempel 4. diagnosen hvis kontekst der prædikteres er "anteroseptal myocardial infarction".

## 4.2 SNOMED-CT (ShARe)

Da strict og relaxed F-score var identiske i denne evaluering, vil der kun optræde én F-score. Systemet viste at kunne prædiktere størstedelen af termerne. Modellen havde lige mange falsk negative som falsk positive prædiktioner. På tabel 4.3 er fejldetekteringerne vist.

| | Precision | Recall | F |
|---|---|---|---|
| ShARe Dataset | 0.98 | 0.98 | 0.98 |

**Tab. 4.3.:** Resultater for Share datasættet. Der var 101 sand positive, 3 falsk negative og 3 falsk positiv.

Falsk positive eksempler er annoterede med FP, og falsk negative eksempler er annoteret som FN. Falsk negative eksempler er markeret med grøn og blå, hvor rød angiver at det fundne term er falsk positiv.

På figur 4.12 var den diagnose der skulle detekteres "slowing heart rate". Der er ikke blevet annoteret nogle diagnoser for ord der er adskilte af ikke annoterede ord ("of the"), og dette kan muligvis forklare fejldetekteringen.

since the previous tracing of _date minimal slowing of the heart rate FN has occurred.
no other significant changes are seen.

**Fig. 4.12.:** Eksemplet viser et falsk negativt eksempel.

På figur 4.13 var de diagnoser der skulle detekteres "atrial pacemaker artifact"og “pacemaker capture”. Det blev set at “pacemaker capture” er blevet annoteret som en diagnose, når der står “pacemaker without capture”, hvilket muligvis kunne være en fejl i annoteringen.

sinus rhythm and underlying sinus rhythm without ventricular sensing of the atrial
pacemaker. there is intermittent appearance of apparent atrial pacemaker artifact FN
without capture. FN clinical correlation is suggested.

**Fig. 4.13.:** Eksemplet viser et falsk negativt eksempel.

På figur 4.14 kan det ses at “atrial premature depolarization” blev detekteret af NER-algoritmen, men den kunne ikke mappes korrekt til en SNOMED CT kode, da denne skulle mappes til "ectopic atrial beats".

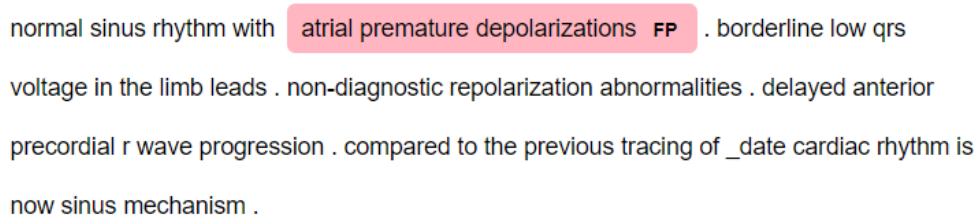


**Fig. 4.14.:** Eksemplet viser et falsk positivt eksempel.

# 5 Syntese

## 5.1 Diskussion

*I dette kapitel vil resultaterne for den inkrementelle model først diskuteres, herunder de tre enkeldele; NER, kontekst klassifikation, og evaluering på ShARe datasættet. Derefter følger overvejelser i forhold til resultaterne fra prædiktionerne på de enkelte eksempler. Derudover vil den kliniske betydning for systemet blive diskuteret. Efterfølgende diskuteres begrænsninger, og eventuelle forbedringer til det foreslåede system. Slutteligt vil potentialet for det udviklede system diskuteres, herunder overvejelser i forhold til fremtidsmuligheder for systemet.*

### 5.1.1 Algoritmernes performance

Resultaterne for NER-algoritmen viste at modellen havde en F1-score på 0.89 på testsættet, med en recall på 0.91 og en precision på 0.88. Dette betyder at modellen har et højere antal falsk positive end falsk negative. Det er dog i denne forbindelse vigtig at understrege at F1 score, recall og precision alle er beregnede for eksakte match. Dette skal ses i lyset af at den gennemsnitlige længde for navnet på en diagnose var 2.9 ord. Navnet for en diagnose skulle altså detekteres 100 procent korrekt i hele sin længde, for at blive medregnet som et match, hvorfor man kan argumentere for at algoritmen kan have detekteret en diagnose meget tæt på korrekt, uden at tilfældet er blevet medtaget i statistikken for præcision. Det blev observeret, at selvom algoritmen kunne håndtere fejl eller ændringer i enkelte ord i en diagnose, så opstod der fejldetekteringer når der var fejl i flere ord i samme diagnose. Dette kunne f.eks. være stavefejl i en diagnose der bestod af flere ord. Dette omhandlede dog ikke kun stavefejl, men også fejl i tokenization, hvis en anden fejl allerede var til stede. Dette blev observeret ved f.eks. "a - v coondoction", hvor "a - v", var opdelt i tre tokens, men skulle have været én token, da der er tale om en forkortelse for "atrioventricular". Der har ikke været nogle studier der sammenligner evnen til at finde kandidatbegreber i EKG-notater, men i et studie af Wu m.fl. [61] blev den samme LSTM model benyttet, som blev benyttet i dette projekt. I nævnte studie blev også benyttet word embeddings der var trænet på MIMIC-corpus, dog uden de benyttede character embeddings, som blev anvendt i dette projekt. Derudover arbejdede de ikke med samme datasæt som anvendt i dette projekt, men

formål samt evalueringsmetode minder om fremgangsmetoden i dette projekt. I studiet opnåede de en F1-score på 85.94%, hvilket er den højst rapporterede score for det anvendte datasæt. Karakteristisk for deres studie var, at de opnåede en højere recall end precision Wu m.fl. [61] og dette kunne indikere at denne type neuralt netværk har en tendens til at overdetektere de entiteter der skal findes. Da kandidatbegreber efterfølgende skal mappes til en unik kode, via en anden algoritme, vil dette fungere som en sikringsmekanisme, da den anden algoritme forhindrer at de fejldetekterede kandidatbegreber slipper gennem nåleøjet. Fælles for begge algoritmer var, at der kunne opstå tvivl i forhold til hvordan eksempler skulle annoteres. For NER-algoritmen, var det nogle gange vanskeligt at bestemme grænserne for hvor ord startede og sluttede, og for kontekstklassifikationsmodellen, var der subtile negeringer eller ubekræftede diagnoser, hvor der var tvivl om hvad der var den rigtige kategori til kandidatbegrebet.

F1-scoren for kontekstklassifikationen var i dette projekt på 0.92, hvilket er sammenligneligt med studiet af Chen [5], hvor metoden til at markere kontekst ord i dette projekt stammer fra. I studiet sås en F1-score fra 0.88-0.93. Der skal dog tages højde for at de ikke er testet på samme datasæt, og derfor vil en præcis sammenligning ikke kunne udføres. Det blev desuden observeret at der i testsættet anvendt i dette projekt, var relativt få negerede eksempler til stede, hvilket kan skabe usikkerhed omkring den reelle styrke på algoritmen. Derudover var der tre kontekstklasser i dette system, i modsætning til to i Chen [5]. Generelt skal machine learning modeller gentrænes, hvis der benyttes andre datasæt end det datasæt modellen oprindeligt blev trænet på. Det kan dog formodes at dele af konktekstklassifikationsmodellen kan genbruges, ved at fintune modellen til andre engelsksprogede kliniske datasæt. Det blev ligeledes observeret at der kunne forekomme et overlap mellem negerede og ubekræftede diagnoser. Dette kan vise sig at være problematisk ved benyttelse af softmax aktiveringsfunktionen, som det blev benyttet i kontekstklassifikationsmodellen. Denne aktiveringsfunktion vil favorisere en bestemt klasse, og dette kan være uhensigtsmæssigt, hvis en diagnose faktisk tilhører flere forskellige klasser. Her vil det være mere hensigtsmæssigt at benytte aktiveringsfunktioner der udregner sandsynligheden for klasserne individuelt, og derefter finde en tærskelværdi for hvornår et bestemt output medfører at en diagnose tilhører en bestemt klasse.

I evalueringen af EKG-notaterne på shARe datasættet blev der opnået en precision, recall og strict F-score på henholdsvis 0.98, 0.98 og 0.98, hvilket betyder at algoritmerne kunne detektere langt de fleste diagnoser og mappe dem korrekt til en SNOMED CT kode. Dette tal kan virke højt, men de fleste eksempler var trivielle. De fejl der var, viser dog en problematik med sådanne systemer, idet de synliggør at algoritmen har svært ved at detektere termer der ikke ligger i umiddelbar forlængelse af hinanden. Eksempler på dette kunne f.eks. være vanskeligheder ved detektering af en diagnose i sætningen "slowing of the heart rate", hvor "slowing heart rate"ville

have været det korrekte span. Overordnet set er det svært at vurdere hvordan modellerne performer i mødet med denne opgave, i forhold til andre systemer, da der ikke findes systemer hvor én-til-én sammenligning kan foretages. Dog blev der i studiet af Denny m.fl. [12] rapporteret en precision og recall på 0.93 og 0.93, på et system der bl.a. kunne mappe ECG-findings til UMLS koder. Dette regelbaserede system adskilte sig dog ved at EKG-noterne var maskingenererede, og dermed var der f.eks. ikke behov for at tage højde for stavefejl og forskellige varianter af samme ord. Derudover kunne systemet heller ikke detektere konteksten for et begreb. To konkurrencer der har benyttet samme EKG-notater i deres datasæt, som dette projekt anvender, er SemEval 2014 task 7 og SemEval 2015 task 14. Her var de benyttede EKG-notater en mindre del af det samlede datasæt. Ved sammenligning ses at dette projekts system har en højere strict og relaxed F-score end disse. I disse konkurrencer er den højeste F-score på 0.87. Desuden bør en række forbehold tages. I de to konkurrencer var EKG-notaterne inkluderet som træningsdata, og ikke som testdata, og derudover inkluderede datasættene også andre notattyper som f.eks. radiologirapporter og epikrisenotater.

### 5.1.2 Effektiv dataindsamling

Active learning delen i dette projekt, har vist sig at kunne fungere. Dog ville man, for at kunne evaluere om metoden har været bedre end at udvælge eksempler tilfældigt, som er den tradionelle metode, være nødt til at skulle annotere data både med active learning metoden, og med den traditionelle metode. Da formålet med projektet var at konstruere et system, og ikke at teste om active learning er brugbart i forhold til named-entity-recognition, blev dette ikke valgt. Dog er det relevant at understrege at studiet [29, 50] fandt at active learning metoden var mest effektiv.

For at kunne implementere systemet vil det være nødvendigt at udføre grundige brugertests, hvor klinisk personale vil skulle evaluere det foreslåede system i forhold til brugervenlighed og effektivitet. Det vil derudover være et krav at udføre flere test af algoritmernes nøjagtighed. Derudover vil det kræve at feedbackmodulet evalueres grundigt, for dermed at kunne klarlægge på hvilke tidspunkter systemet ville gavne af at blive gentrænet. Det blev observeret, at der var få inkonsistente annoteringer, hvilket kan have medført et fald i accuracy. Der kan stilles spørgsmåltegn ved hvorvidt en NER-agoritme med flere klasser havde været fordelagtig. I dette projekt blev det forsøgt at benytte en algoritme der kunne detektere kandidatbegreber og kontekst på samme tid. Dette virkede, men ikke lige så godt som to separate algoritmer. Trenden i NLP er at benytte sig af store pretrænede modeller, hvor algortimerne fintunes til forskellige opgaver. I de seneste år har andre modelarkitekturer gjort deres indtog. I denne forbindelse kan for eksempel nævnes transformer arkitekturen, som muliggør parralisering, imodsætning til RNN modeller som ikke har denne mulighed.

Disse modeller vil højst sandsynlig blive mere prominente i fremtidens kliniske NLP. Slutteligt skal det nævnes at der er ikke taget højde for postkoordinerede udtryk i arbejdet med dette projekt, hvilket vil være hensigtsmæssigt ved en reel implementering.

## 5.2 Konklusion

I dette projekt blev problemstillingen besvaret: "*Hvordan kan neurale netværk anvendes til at detektere diagnoser og deres kontekst i EKG-notater, når der skal tages højde for at mængden af annoteret data til klinisk NLP er begrænset?* "

I projektet blev det fundet, at det kunne lade sig gøre at konstruere et neuralt netværk, der vha. af forskellige teknikker kunne benyttes til at detektere diagnoser. Det blev fundet, at den bedste model der kunne skabes til denne opgave, var et long short-term memory (LSTM) netværk, med inddragelse af character embeddings. Det viste sig at være vanskeligt at finde tilsvarende studier til sammenligning, dog kan blandt andre nævnes studiet af Wu m.fl. [61], der også anvendte en LSTM model, og opnåede en lignende F1 score. Studiet af Wu m.fl. [61] anvendte dog ikke character embeddings, og anvendte heller ikke samme datasæt, hvorfor en fuldstændig sammenligning ikke er mulig. I dette projekt blev anvendt en active learning strategi, der ved hjælp af en NER model med character embeddings, udvalgte de mest informative eksempler i en mængde ikke-annoteret tekst. Derudover blev en semi-superviseret model, som var trænet på store mængder ikke-annoteret tekst, foreslået til at detektere konteksten for diagnoser. Denne model viste sig at have stort potentiale, i forhold til en præcis detektering af en bestemt diagnoses kontekst, uden at være blevet trænet på store mængder annoteret data. Denne kombination af metoder, er hvad der adskiller dette projekt fra andre lignende projekter, og det er denne kombination der tillader at modellen kan operere med høj sikkerhed, selvom der kun var en relativt lille mængde annoteret data tilgængeligt. Der blev desuden foreslået en metode til at forbedre systemet efter implementering, ved at benytte klinikeren til at bekræfte eller tilrette systemets foreslåede prædiktioner, hvorefter systemet vil lære af disse bekræftelser eller rettelser, og yderligere øge sin præcision. Hvis diagnosedetektering på EKG-notater implementeres i praksis, vil de største potentialer være af både tidsbesparende og kvalitativ signifikans. Klinikerene vil spare tid i forhold til søgning i EKG-notaterne, og med over 300.000.000 EKG'er om året, alene i USA, skal denne mulige tidsbesparelse altså ses på en omfattende skala. Desuden vil systemet kunne mindske risikoen for at vigtige aspekter af en patients sygdomshistorik overses, f.eks. i forbindelse med senere behandling. Den ekstra tid som frigives for klinikeren, i forhold til søgning i dokumenterne, vil desuden kunne bruges med patienten, hvilket yderligere kan øge kvaliteten af behandlingen. Der er altså vidtrækkende muligheder i forbindelse med en implementering af systemet, hvorfor dette projekt finder at yderligere udforskning af potentialet vil være hensigtsmæssigt.

# Annoteringsvejledning

# A

Målet for dette projekt er at detektere disorders. En disorder kan bestå af flere ord eller tegn. Positionerne fra start til slut af et disorder term, benævnes span. I dette projekt anvendes ordet “disorder”, som beskrevet i SNOMED-CT’s definition af begreber som tilhører hierarkiet “Disorder semantic group”. Disorder hierarkiet er en del af clinical finding hierarkiet. Disorder begreber skal have en forældre af typen disorder, og ikke clinical finding. Ved en disorder gælder det at den altid er udtryk for en abnormal tilstand, og derudover har en disorder også altid en underliggende patologisk proces.

Derudover gælder det også at en disorder skal annoteres, hvis det anvendte ord er et af flere synonymer for den samme disorder . Der søges at detektere den mest specifikke beskrivelse af en given disorder. Dette betyder f.eks. at i sætningen “The patient has a small bowel obstruction”, vil den mest præcise disorder beskrivelse være “small bowel obstruction” og ikke “bowel obstruction”. Et begreb skal have en SNOMED-CT kode tilknyttet, og hvis denne ikke kan findes, men begrebet stadig lever op til kriterierne for at være en disorder, skal denne annoteres som CUI-less.

Derudover annoteres negerede disorders. Dette betyder at disorderne ikke forefindes. Et eksempel på dette er: “ The patient does not have cancer”. Derudover ønskes at detektere usikre diagnoser. F.eks. “There has been possible pain.” [17]

# LSTM

# B

Long short-term memory netværk er en afart af en bestemt type neurale netværk(Recurrent neural network). Denne type af netværk, kan lære længere tidsafhængigheder end traditionelle neurale netværk. LSTM netværket består af en række enheder, med såkaldte "gates", der kan styre informationsflowet fra unit til unit. Dette betyder, at denne type netværk har lettere ved at fokusere på specifikke dele af inputtet, og dermed er det disse der er med til at give netværket "hukommelse". En unit tager $x_t$, $h_{t-1}$ og $c_{t-1}$ som input. $x$ er ordvektoren, $h$ er en units hiddenstate og $c$ er en units cell state. $\sigma$ er en sigmoid funktion, og W er de fire vægtmatricer, og det er disse matricer der optimeres ved hjælp af gradient descent. $b$ er bias vektorerne.

$$i_i = \sigma(Wx_{xi}x_t + W_{hi}h_{t-1} + W_{c1}c_{t-1}b_i) \tag{B.1}$$

$$f_t = \sigma(Wx_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \tag{B.2}$$

$$c_t = f_t \bigodot c_{t-1} + i_t \bigodot tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{B.3}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \tag{B.4}$$

$$o_t \bigodot tanh(c_t) \tag{B.5}$$

# Kodeeksempler

C

## Embeddings

```
import gensim
from pathlib import Path
from flair.data import Dictionary
from flair.models import LanguageModel
from flair.trainers.language_model_trainer import LanguageModelTrainer, TextCorpus

class Create_wordvectors():
    '''
    Class that creates w2v and character embeddings

    Args:
        datafile (str): The path for the textcorpus used for creation of the embeddings

    '''
    def __init__(self, datafile):

        self.datafile = datafile


    def _w2vtxt_processer(self, inputfile):
        with open(inputfile, 'rb') as f:
            for i, line in enumerate (f):
                yield gensim.utils.simple_preprocess (line)

    def __call__(self,**kwargs):

        if 'word2vec' in kwargs:
            params = kwargs['word2vec']
            documents = list (self._w2vtxt_processer(self.datafile))
            w2vmodel = gensim.models.Word2Vec (documents, size=params['size'],
                                               min_count=params['min_count'],
                                               negative=params['negative'])
            w2vmodel.train(documents,total_examples=len(documents),epochs=10)
            w2vmodel.save('w2v.gensim')

        if 'charemb' in kwargs:
            params = kwargs['charemb']
            is_forward_lm = params['forward']
            dictionary = Dictionary.load('chars')
```

```
            corpus = TextCorpus(Path('C:/Users/Martin/corpus'),
                    dictionary,
                    is_forward_lm,
                    character_level=True)

            language_model = LanguageModel(dictionary,
                                is_forward_lm,
                                hidden_size=params['hidden_size'],
                                nlayers=1)
            # train your language model
            trainer = LanguageModelTrainer(language_model, corpus)
            trainer.train('resources/taggers/lang',
              sequence_length=params['sequence_length'],
              mini_batch_size=10,
              max_epochs=10)

data_file='ecgcorpus.txt'
wv = Create_wordvectors(data_file)
wv(word2vec= {'size':100,'min_count':2,'negative':20})
wv(charemb= {'forward':True,'hidden_size':128,'sequence_length':100})
```

## Træning af NER algoritme

```
from flair.models import SequenceTagger
from flair.embeddings import TokenEmbeddings, WordEmbeddings
from flair.embeddings import StackedEmbeddings, FlairEmbeddings
from flair.data import TaggedCorpus
from flair.data_fetcher import NLPTaskDataFetcher

# define columns
columns = {0: 'text', 1:'ner'}

data_folder = 'C:/Users/Martin/Desktop/Notebooks/speciale_ner/finaldataner/'
# retrieve corpus using column format, data folder
# and the names of the train, dev and test files
corpus: TaggedCorpus = NLPTaskDataFetcher.load_column_corpus(data_folder, columns,
                                                             train_file='train.txt',
                                                             test_file='test.txt',
                                                             dev_file='val.txt')

lm_embeddings = FlairEmbeddings('resources/taggers/language_model/best-lm.pt')
char_lm_embeddings2 = FlairEmbeddings('resources/taggers/language_modelb/best-lm.pt')
custom_embedding = WordEmbeddings('C:/Users/Martin/Desktop/Notebooks/speciale_ner/w2v.gensim')

tag_type = 'ner'
tag_dictionary = corpus.make_tag_dictionary(tag_type=tag_type)

embedding_types = [custom_embedding, char_lm_embeddings,char_lm_embeddings2]

```

```
embeddings = StackedEmbeddings(embeddings=embedding_types)



tagger = SequenceTagger(hidden_size=128,  # lstm by deaf. go into source for GRU
                                          embeddings=embeddings,
                                          tag_dictionary=tag_dictionary,
                                          tag_type=tag_type,
                                          use_crf=True,rnn_layers=2, dropout=0.2)


trainer = ModelTrainer(tagger, corpus)

trainer.train('resources/taggers/example-1nerthhw2',
              learning_rate=0.1,
              mini_batch_size=16,
              max_epochs=50,
             checkpoint=True
            )
```

# Træning af UlmFit

```
import pandas as pd

## Create markers for the context word
def add_row(dataset, text, target):
    dataset = dataset.append({'text':text, 'target':target},ignore_index=True)
    return dataset



def sentence_2_df(df,sentence):
    sent = sentence
    spans_in_txt = []
    for span in sent.get_spans('ner'):
        idx = [token.idx for token in span.tokens]
        spans_in_txt.append((idx,span.tag))

    marker = ""
    index = len(sent.get_spans('ner'))

    for idx in range(index):
        marker = ""
        for i in range(1,len(sent)+1):
            if spans_in_txt[idx][0][0]==i:
                marker += "xxstart " # instead of <c>
            elif spans_in_txt[idx][0][-1]==i-1:
                marker += "xxslut " # instead of <\c>
            val = sent.get_token(i)
```

```
            marker += val.text + ' '

        if spans_in_txt[idx][1] != "":
            df = add_row(df, marker, spans_in_txt[idx][1])
    return df

traindev = pd.DataFrame() # Cre
for test in corpus_context.train:
    traindev = sentence_2_df(df, test)

for test in corpus_context.def:
    traindev = sentence_2_df(df, test)

traindev.to_csv('traindev.csv',index=False)

path = 'C:/Users/Martin/Desktop/Notebooks/'

bs = 32
data_lm = load_data(path, 'data_lm.pkl', bs=bs)

data_clas = (TextList.from_df(data_for_clas, vocab=data_lm.vocab,cols='text')
             .split_by_rand_pct(0.15,seed=17)
             .label_from_df(classes=['POS_DIS', 'NEG_DIS','UN_DIS'],cols='target')
             .databunch(bs=bs))
# WikiModel replaced by MIMIC-model
# Target task language model fine-tuning and TTCF is omitted ion this example
learn = text_classifier_learner(data_clas, AWD_LSTM, drop_mult=0.5,pretrained=True)
learn.load_encoder('fine_tuned_enc_final') # ecg corpus model
learn.lr_find()
learn.recorder.plot()
learn.fit_one_cycle(1, 1e-2)
learn.freeze_to(-2)
learn.fit_one_cycle(1, slice(1e-2/(2.6**4),1e-2))
learn.freeze_to(-3)
learn.fit_one_cycle(1, slice(5e-3/(2.6**4),5e-3), moms=(0.8,0.7))
learn.unfreeze()
learn.fit_one_cycle(2, slice(1e-3/(2.6**4),1e-3), moms=(0.8,0.7))
learn.save('sent_classifier') #
learn.load('sent_classifier')
preds, targets = learn.get_preds(ds_type=DatasetType.Test, ordered=True)
predictions = np.argmax(preds, axis = 1)
pd.crosstab(predictions, targets)
```

## Utils

```
## AL . prodi.gy
import time
from flair.data import Sentence
import pandas as pd
```

```
from sklearn.utils import shuffle
#use prodi.gy instead
tagger: SequenceTagger = SequenceTagger.load_from_file('/best-model.pt')
dfObj = pd.read_csv('datas')
dfObj = dfObj.loc[dfObj['used']==0]
dfObj = shuffle(dfObj)
anready = []

exam = 300
for i in range(exam):
    sentence = Sentence(dfObj['text'][i])
    tagger.predict(sentence)
    props = 1
    if len(sentence.tokens)>1:
        for i in range(1,len(sentence.tokens)):
            #print(sentence.get_token(i).tags['ner'].score)
            props *= sentence.get_token(i).tags['ner'].score
            #print(sentence.get_token(i).tags['ner'].score)
        tot_sum =  props #normalize  props/(len(sentence.tokens)) #normalize
        anready.append((dfObj['text'][i],1-.0tot_sum))
        dfObj['used'][i] = 1


o = [i for i,j in anready]
p = np.array([j for i,j in anready])
p /= p.sum()
np.random.choice(o, 250, p=p, replace=False)

.
.
.

def sieve_algo(text):
    a = exact_macth(text)
    if a == 'NONE':
        a = hyphenation(text)
    if a == 'NONE':
        a = abreviation(text)
    if a == 'NONE':
        a = norepl(text)
    if a == 'NONE':
        a = partialmatch(text)
    return a


.
.
.
# preproc
ecgreports_df['TEXT'] = ecgreports_df['TEXT'].str.replace('\[\*\*\d+\S+', '_date', regex=True)
ecgreports_df['TEXT'] = ecgreports_df['TEXT'].str.replace('\[\*\*Month.*\]', '_date', regex=True)
```

```
ecgreports_df['TEXT'] = ecgreports_df['TEXT'].str.replace('\[\*\*\d+\S+', '_date', regex=True)
ecgreports_df['TEXT'] = ecgreports_df['TEXT'].str.replace('\[\*\*Last.*\]', '_name', regex=True)
ecgreports_df['TEXT'] = ecgreports_df['TEXT'].str.replace('\[\*\*Name.*\]', '_name', regex=True)
ecgreports_df['TEXT'] = ecgreports_df['TEXT'].str.replace('\[\*\*Doctor.*\]', '_doctor', regex=True)
ecgreports_df['TEXT'] = ecgreports_df['TEXT'].str.replace('\n([A-Z])', '. \\1', regex=True)
ecgreports_df['TEXT'] = ecgreports_df['TEXT'].str.replace('\n', ' ', regex=True)
ecgreports_df['TEXT'] = ecgreports_df['TEXT'].str.replace(' +', ' ', regex=True)
ecgreports_df['TEXT'] = ecgreports_df['TEXT'].str.replace('\.+', '.', regex=True)
ecgreports_df['TEXT'] = ecgreports_df['TEXT'].str.replace('\[.+\]', '_unknown', regex=True)
ecgreports_df['TEXT'] = ecgreports_df['TEXT'].str.lower()
```

# Bibliografi


[1]J Bergstra og R Bardenet. „Algorithms for Hyper-Parameter Optimization“. I: (), s. 1–9. arXiv: 1206.2944 (se s. 28).

[2]Raymond R Bond, Dewar D Finlay, Chris D Nugent og George Moore. „A review of ECG storage formats“. I: *International Journal of Medical Informatics* 80.10 (2011), s. 681–697 (se s. 4).

[3]Sergej Bondar, John C. Hsu, Alain Pfouga og Josip Stjepandić. „Agile Digitale Transformation of Enterprise Architecture Models in Engineering Collaboration“. I: *Procedia Manufacturing* 11 (2017), s. 1343–1350 (se s. 3).

[4]Barry D Brown, Fabio Badilini og A M P S. „HL7 aECG Implementation Guide Principal Contributors :“ i: (2005) (se s. 4, 5).

[5]Long Chen. „Attention-Based Deep Learning System for Negation and Assertion Detection in Clinical Notes“. I: *International Journal of Artificial Intelligence & Applications* 10.01 (2019), s. 1–9 (se s. 6, 49).

[6]Long Chen. „Attention-Based Deep Learning System for Negation and Assertion Detection in Clinical Notes“. I: *International Journal of Artificial Intelligence & Applications* 10.01 (2019), s. 1–9 (se s. 30).

[7]Yukun Chen, Thomas A Lasko, Qiaozhu Mei, Joshua C Denny og Hua Xu. „A study of active learning methods for named entity recognition in clinical text“. I: *Journal of Biomedical Informatics* 58 (2015), s. 11–18 (se s. 24).

[8]Junyoung Chung, Caglar Gulcehre, KyungHyun Cho og Yoshua Bengio. „Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling“. I: (2014), s. 1–9. arXiv: 1412.3555 (se s. 27).

[9]Susan J. Curry, Alex H. Krist, Douglas K. Owens m.fl. „Screening for cardiovascular disease risk with electrocardiography us preventive services task force recommendation statement“. I: *JAMA - Journal of the American Medical Association* 319.22 (2018), s. 2308–2314 (se s. 3).

[10]Joshua C. Denny. „Mining Electronic Health Records in the Genomics Era“. I: *PLoS Computational Biology* 8.12 (2012) (se s. 4).

[11]Joshua C. Denny og Anderson Spickard. „Identifying UMLS concepts from ECG Impressions using KnowledgeMap“. I: (2008), s. 73 (se s. 8).

[12]Joshua C Denny, Randolph A Miller, Lemuel Russell, Mark A Arrieta og Joshua F Peterson. „Identifying QT prolongation from ECG impressions using a general-purpose Natural Language Processor“. I: (2008), s. 34–42 (se s. 1, 5, 50).

[13]Joshua C Denny, Jeffrey D Smithers, Randolph A Miller og Anderson Spickard. „”Understanding” Medical School Curriculum Content Using KnowledgeMap“. I: 10.4 (2003), s. 351–362 (se s. 8).

[14]Jennifer D'Souza og Vincent Ng. „Sieve-Based Entity Linking for the Biomedical Domain“. I: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)* (2015), s. 297–302 (se s. 34).

[15]Computer-assisted Electrocardiography, Paul Rubel, Danilo Pani m.fl. „SCP-ECG V3 . 0 : An Enhanced Standard Communication Protocol for“. I: 43.November 2013 (2016), s. 1–4 (se s. 5).

[16]N. Elhadad, S. Pradhan, S. L. Gorman m.fl. „SemEval-2015 Task 14 : Analysis of Clinical Text“. I: *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)* January (2015), s. 303–310 (se s. 9).

[17]Noemie Elhadad, Guergana Savova, Wendy Chapman m.fl. „ShARe Guidelines for the Annotation of Modifiers for Disorders in Clinical Notes“. I: January 2011 (2012), s. 1–36 (se s. 53).

[18]Euan A Ashley og Josef Niebauer. *Cardiology Explained,Remedica Explained Series*. 2004 (se s. 34).

[19]Https://prodi.gy/ Explosion AI. „No Title“. I: *https://prodi.gy* () (se s. 24).

[20]Carlos R. García-Alonso, Leonor M. Pérez-Naranjo og Juan C. Fernández-Caballero. „Visualizing Data using t-SNE“. I: *Journal of Machine Learning Research* 9 (2008), s. 2579–2605. arXiv: 1307.1662 (se s. 25).

[21]Sebastian Gehrmann, Franck Dernoncourt, Yeran Li m.fl. „A Comparison of Rule-Based and Deep Learning Models for Patient Phenotyping“. I: () (se s. 6).

[22]Maximilian Hofer, Andrey Kormilitzin, Paul Goldberg og Alejo Nevado-Holgado. „Few-shot Learning for Named Entity Recognition in Medical Text“. I: (2018), s. 1–10. arXiv: 1811.05468 (se s. 10).

[23]Jeremy Howard og Sebastian Ruder. „Universal Language Model Fine-tuning for Text Classification“. I: (2018). arXiv: 1801.06146 (se s. 30, 32).

[24]Https://www.fast.ai/. „FastAI“. I: () (se s. 37).

[25]Peter B. Jensen, Lars J. Jensen og Soøren Brunak. „Mining electronic health records: Towards better research applications and clinical care“. I: *Nature Reviews Genetics* 13.6 (2012), s. 395–405 (se s. 1, 5–7).

[26]Alistair E.W. Johnson, Tom J Pollard, Lu Shen m.fl. „MIMIC-III, a freely accessible critical care database“. I: *Scientific Data* 3 (2016), s. 160035 (se s. 15).

[27]Rafal Jozefowicz og Ilya Sutskever. „An Empirical Exploration of Recurrent Network Architectures Rafal“. I: *International Conference on Machine Learning* (2015) (se s. 27).

[28]Mahnoosh Kholghi. „Active Learning for Concept Extraction from Clinical Free Text“. Ph.d.-afh. Queensland University of Technology, 2017 (se s. 10).

[29]Mahnoosh Kholghi, Laurianne Sitbon, Guido Zuccon og Anthony Nguyen. „Active learning: a step towards automating medical concept extraction“. I: *Journal of the American Medical Informatics Association* 23.2 (2016), s. 289–296 (se s. 24, 50).

[30]Mahnoosh Kholghi, Laurianne Sitbon, Guido Zuccon og Anthony Nguyen. „International Journal of Medical Informatics“. I: *International Journal of Medical Informatics* 106.August (2017), s. 25–31 (se s. 24).

[31]Tsung-Ting Kuo, Pallavi Rao, Cleo Maehara m.fl. „Ensembles of NLP Tools for Data Element Extraction from Clinical Notes“. I: *AMIA Annual Symposium* 1 (2016), s. 1880–1889 (se s. 7).

[32]Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami og Chris Dyer. „Neural Architectures for Named Entity Recognition“. I: (2016). arXiv: 1603.01360 (se s. 22, 28).

[33]Zengjian Liu, Ming Yang, Xiaolong Wang m.fl. „Entity recognition from clinical texts via recurrent neural network“. I: *BMC Medical Informatics and Decision Making* 17.Suppl 2 (2017) (se s. 9, 10).

[34]Yen-fu Luo, Weiyi Sun og Anna Rumshisky. „A Hybrid Method for Normalization of Medical Concepts in Clinical Narrative“. I: *2018 IEEE International Conference on Healthcare Informatics (ICHI)*. IEEE, 2018, s. 392–393 (se s. 10).

[35]Joel Martin, Xiaodan Zhu, Berry de Bruijn, Svetlana Kiritchenko og Colin Cherry. „Machine-learned solutions for three stages of clinical information extraction: the state of the art at i2b2 2010“. I: *Journal of the American Medical Informatics Association* 18.5 (2011), s. 557–562 (se s. 9).

[36]Marvelapp.com. „Marvelapp“. I: () (se s. 14).

[37]Paul Mele. „Improving electrocardiogram interpretation in the clinical setting“. I: *Journal of Electrocardiology* 41.5 (2008), s. 438–439 (se s. 3).

[38]Stephen Merity, Nitish Shirish Keskar og Richard Socher. „Regularizing and Optimizing LSTM Language Models“. I: (2017). arXiv: 1708.02182 (se s. 30).

[39]S. M. Meystre. „Extracting Information from Textual Documents in the Electronic Health Record: A Review of Recent Research“. I: *Physical Review B* 95.20 (2017), s. 128–144 (se s. 1, 5, 6).

[40]Tomas Mikolov, Kai Chen, Greg Corrado og Jeffrey Dean. „10.1162/Jmlr.2003.3.4-5.951“. I: *CrossRef Listing of Deleted DOIs* 1 (2006), s. 1–9. arXiv: 1806.06259 (se s. 22).

[41]Danielle L. Mowery, Sumithra Velupillai, Brett R. South m.fl. „Task 2: ShARe/CLEF eHealth evaluation lab 2014“. I: *CEUR Workshop Proceedings* 1180 (2014), s. 31–42 (se s. 15).

[42]Avital Oliver, Augustus Odena, Colin Raffel, Ekin D. Cubuk og Ian J. Goodfellow. „Realistic Evaluation of Deep Semi-Supervised Learning Algorithms“. I: Nips (2018). arXiv: 1804.09170 (se s. 30).

[43]Parth Pathak, Pinal Patel, Vishal Panchal m.fl. „ezDI : A Supervised NLP System for Clinical Narrative Analysis“. I: *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)* SemEval (2015), s. 412–416 (se s. 9).

[44]Jakub Piskorski og Roman Yangarber. „Information Extraction : Past , Present and Future“. I: (), s. 23–50 (se s. 6).

[45]Sameer Pradhan, Noémie Elhadad, Wendy W. Chapman, Suresh Manandhar og Guergana Savova. „SemEval-2014 Task 7: Analysis of Clinical Text“. I: *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)* SemEval (2014), s. 54–62 (se s. 9).

[46]Sameer Pradhan, Noemie Elhadad, Brett R South m.fl. „Task 1 : ShARe / CLEF eHealth Evaluation Lab 2013“. I: (2013), s. 1–6 (se s. 9).

[47]Radim Rehurek. „Gensim“. I: *https://radimrehurek.com/gensim/about.html* () (se s. 22).

[48]S Trent Rosenbloom, Joshua C Denny, Hua Xu m.fl. „Data from clinical notes : a perspective on the tension between structure and flexible documentation“. I: (2011), s. 181–186 (se s. 5, 6).

[49]U.S. Department of Health & Human Services. „No Title“. I: *https://github.com/HHS/uts-rest-api* () (se s. 34).

[50]Yanyao Shen, Hyokun Yun, Zachary C Lipton, Yakov Kronrod og Animashree Anandkumar. „Deep Active Learning for Named Entity Recognition“. I: (2017), s. 1–15. arXiv: 1707.05928 (se s. 13, 24, 50).

[51]Hansen PS. Sigurd BM, Pehrson S. „No Title“. I: *Alment om EKG: Klinisk elektrokardiologi* FADL forla (2015) (se s. 3).

[52]Christine Sinsky, Lacey Colligan, Ling Li m.fl. „Allocation of Physician Time in Ambulatory Practice: A Time and Motion Study in 4 Specialties“. I: *Annals of Internal Medicine* 165.11 (2016), s. 753 (se s. 5).

[53]Leslie N. Smith. „Cyclical learning rates for training neural networks“. I: *Proceedings - 2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017* April (2017), s. 464–472. arXiv: arXiv:1506.01186v6 (se s. 32).

[54]Dejan Stamenov, Marjan Gusev og Goce Armenski. „Interoperability of ECG standards“. I: *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)* (2018), s. 319–323 (se s. 4).

[55]Weiyi Sun, Anna Rumshisky og Ozlem Uzuner. „Evaluating temporal relations in clinical text: 2012 i2b2 Challenge“. I: *Journal of the American Medical Informatics Association* 20.5 (2013), s. 806–813 (se s. 9).

[56]Weiyi Sun, Anna Rumshisky og Ozlem Uzuner. „Evaluating temporal relations in clinical text: 2012 i2b2 Challenge“. I: *Journal of the American Medical Informatics Association* 20.5 (2013), s. 806–813 (se s. 9).

[57]Buzhou Tang, Yonghui Wu, Min Jiang, Joshua C. Denny og Hua Xu. „Recognizing and encoding disorder concepts in clinical text using machine learning and vector space model“. I: *CEUR Workshop Proceedings* 1179 (2013) (se s. 9).

[58]Özlem Uzuner, Brett R. South, Shuying Shen og Scott L. DuVall. „2010 i2b2/VA challenge on concepts, assertions, and relations in clinical text“. I: *Journal of the American Medical Informatics Association* 18.5 (2011), s. 552–556 (se s. 9).

[59]Roland Vollgraf, Duncan Blythe og Roland Vollgraf. „Contextual String Embeddings for Sequence Labeling“. I: *Proceedings ofthe 27th International Conference on Computational Linguistics* (2018), s. 1638–1649 (se s. 23, 27, 29).

[60]Yanshan Wang, Liwei Wang, Majid Rastegar-Mojarad m.fl. „Clinical information extraction applications: A literature review“. I: *Journal of Biomedical Informatics* 77.November 2017 (2018), s. 34–49 (se s. 7, 9, 10).

[61]Yonghui Wu, Min Jiang, Jun Xu, Degui Zhi og Hua Xu. „Clinical Named Entity Recognition Using Deep Learning Models.“ I: *AMIA ... Annual Symposium proceedings. AMIA Symposium* 2017 (2017), s. 1812–1819 (se s. 48, 49, 52).

[62]Tom Young, Devamanyu Hazarika, Soujanya Poria og Erik Cambria. „Recent Trends in Deep Learning Based Natural Language Processing“. I: (2017), s. 1–32. arXiv: 1708.02709 (se s. 7).

[63]Kai Yu, Zhiheng Huang og Wei Xu. „Bidirectional LSTM-CRF Models for Sequence Tagging“. I: (2011). arXiv: arXiv:1508.01991v1 (se s. 27).

[64]Hua Zhang, Yaoyun and Wang, Jingqi and Tang, Buzhou and Wu, Yonghui and Jiang, Min and Chen, Yukun and Xu, Hua and Zhang, {yaoyun and Xu}. „UTH_CCB: A Report for SemEval 2014 – Task 7 Analysis of Clinical Text“. I: SemEval (2014), s. 802–806 (se s. 9).

# Figurer

# Tabeller