Twitter Sentiment Analysis

Johan Selberg och Johannes Bandgren
Department of Computer Science

Sammanfattning

Abstract.

Nyckelord: keywords

Denna uppsats är skriven som en del av det arbete som krävs för att erhålla en kandidatexamen i datavetenskap. Allt material i denna rapport, vilket inte är mitt eget, har blivit tydligt identifierat och inget material är inkluderat som tidigare använts för erhållande av annan examen.

Johan Selberg

Johannes Bandgren

Godkänd,

Opponent: Sample Name, Test Fun

Handledare: Kerstin Andersson

Examinator: Exam

Tacksägelser

Thanks.

Karlstad Universitet, 24 april 2018 Johan Selberg och Johannes Bandgren

Innehåll

Figurer

Tabeller

Kodavsnitt

Kapitel 1

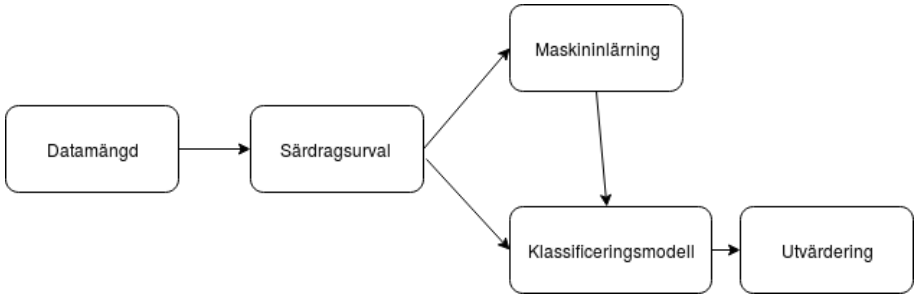
Introduktion

Kapitel 2

Bakgrund

Syftet med studien är att utvärdera tre stycken klassificeringsmodeller inom maskininlärning och hur bearbetningen samt märkningen av en datamängd påverkar en klassificeringsmodells prestanda. Maskininlärningsalgoritmerna som kommer att användas för att få fram klassiciferingsmodellerna är Naive Bayes (NB) och Support Vector Machine (SVM). Den tredje klassificeringsmodellen som tas fram kommer att vara en djuplärande modell som representeras av ett neuronnät, mer specifikt ett Convolutional neural network (CNN). Figur 2.1 visar en förenklad bild av hur processen för att utvärdera klassificeringsmodellerna kommer att se ut. Datamängden består av en mängd indata som är märkt med förväntad utdata, som går igenom en bearbetningsprocess där t.ex. data som inte tillför något värde tas bort och där mängden av data delas upp i tränings- och valideringsdata. Bearbetningsprocessen diskuteras mer ingående i 3.1. På den bearbetade datan utförs ett särdragsurval, där dataspecifika särdrag plockas ut... MER .. som diskuteras vidare i 3.2. Träningsmängden används av maskininlärningsalgoritmen för att skapa klassiciferingsmodellen. När klassificeringsmodellen är framtagen används valideringmängden för att utvärdera prestandan av modellen.

Studien kommer att använda maskininlärning för att utföra sentimentanalys (SA) d.v.s utläsa huruvida en text uttrycker någonting positivt eller negativt. Uppdragsgivaren, CGI, betraktar sentimentanalys som en viktig pusselbit för framtida lösningar man vill erbjuda sina kunder. Lösningar skulle kunna vara chatbotar där



Figur 2.1: Utvärderingsprocessen av maskininlärningsmodeller

sentimentanalys används så att chatboten ändrar sitt språk utefter svaren från slutanvändare. Eller att det kan användas för trendanalys där ett företag vill veta vad allmänheten tycker före och efter att man har släppt en ny produkt eller efter att kvartalsrapporten har släppts.

I avsnitt 2.1 förklaras vad SA är och olika klassificeringstekniker inom SA berskrivs kortfattat. I avsnitt 2.2 ges en kort introduktion till maskininlärning och framstegen som gjorts under den senaste tiden. Dessutom beskrivs NB och SVM samt hur inlärningsprocessen ser ut. I avsnitt 2.3 ges en överblick över hur en datamängd ser ut och vilka märkningsmetoder som används för twittersentimentanalys (TSA). Avsnittet beskriver även de valda datamängderna. I avsnitt 2.4 ges en introduktion till TSA och vilka utmaning samt problem TSA har. Slutligen i avsnitt 2.5 förklaras vilka hjälpmedel som kommer att användas för att utvärdera algoritmernas prestanda.

2.1 Sentimentanalys

SA används för att studera människors åsikter, attityder och känslor mot olika entiteter. En entitet kan vara ett ämne, en händelse eller en individ. Målet med SA är att identifiera känslan som är uttryckt i en text för att därefter analysera den. Processen delas upp i tre steg: att hitta entiteter, identifiera känslan för de entiteterna och slutligen klassificera dessa entiteter. [1].

Inom SA appliceras klassificeringen på 3 olika nivåer: dokument-, menings- och aspektnivå [1]. SA på dokumentnivå klassificerar om ett helt dokument uttrycker en

positiv eller negativ åsikt, exempel på dokument kan vara produktrecensioner eller nyhetsartiklar. Medan på meningsnivå klassificeras varje mening i ett dokument. Slutligen nere på aspektnivå analyseras de möjliga aspekterna av en entitet. En mening kan behandla olika aspekter av en entitet. Både positiva och negativa åsikter kan delges om en entitets olika aspekter. Ett exempel på det är meningen "Ölen var väldigt god, men tyvärr alldeles för dyr", som innehåller både en positiv och en negativ åsikt om en entitet som i det här fallet är ölen.

2.1.1 Klassificeringstekniker

De olika klassificeringsteknikerna som i nuläget används för SA delas upp i 3 olika kategorier: maskininlärningsmetoder, lexikonbaserade metoder samt hybrida metoder [1].

Maskininlärningsmetoder använder etablerade maskininlärningsalgoritmer tillsammans med språkliga särdrag för att konstruera klassificerare som kan avgöra om en text uttrycker någonting positivt eller negativt [ref=Sentiment analysis a survey]. Prestandan för maskininlärningsmetoder är beroende av mängden träningsdata, större mängder data ger vanligtvis bättre resultat [2]. Maskininlärningsmetoder är dessutom domänberoende, vilket gör att de inte presterar bra när de används på andra domäner än det de har tränats på.

Lexikonbaserade metoder använder sig av ordlistor för att analysera text [1]. Ordlistorna består av positiva och negativa termer som används för att beräkna vad en given text uttrycker för sentiment. Fördelen med lexikonbaserade metoder är att de inte kräver någon träningsdata [2]. Men faktumet att de är beroende av statiska ordlistor betyder att de inte tar hänsyn till termer som inte finns i ordlistorna. Det betyder att ordlistor måste uppdateras kontinuerligt för innehåll som är dynamiskt och ständigt under utveckling, vilket är särskilt problematiskt för texter som förekommer på sociala medier.

Hybrida metoder kombinerar lexikonbaserade metoder med maskininlärningsmetoder [2]. Genom att kombinera metoderna med varandra kan de väga upp för varandras svagheter. Nackdelen med hybrida metoder är dock att de kräver en hög beräkningskomplexitet.

Experimentet som presenteras i studien fokuserar enbart på maskininlärnings-

metoder.

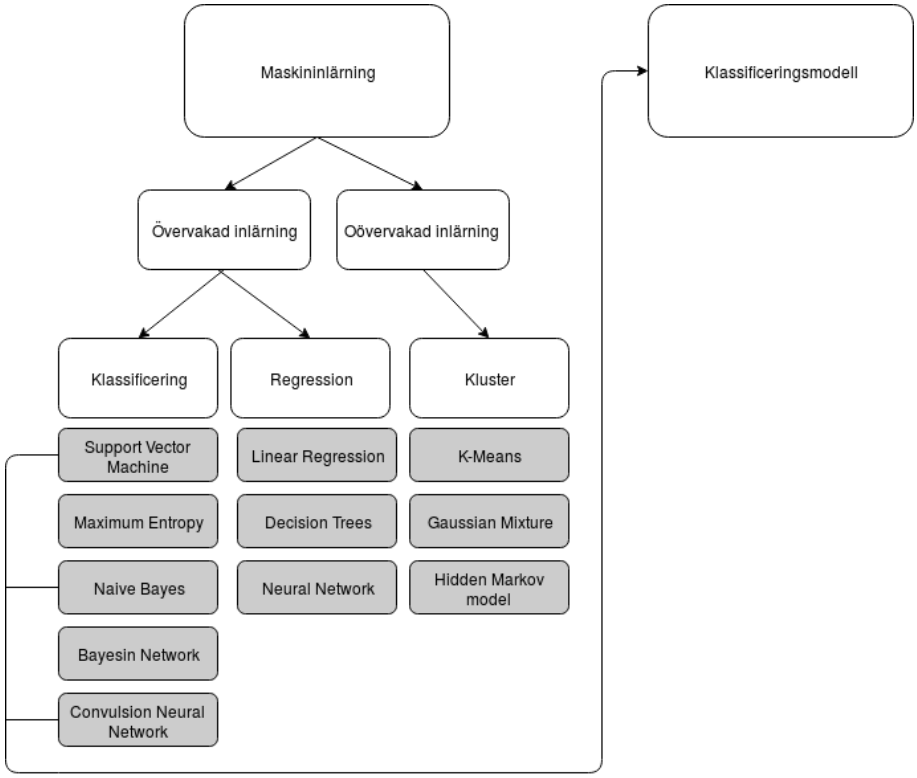
2.2 Maskininlärning

Maskininlärning är ett delområde inom artificiell intelligens (AI), där målet är att göra det möjligt för datorer att lära sig på egen hand. Maskininlärningsalgoritmer gör det möjligt att identifiera olika mönster från observerad data, bygga upp en generell modell som kan förutsäga saker utan att ha blivit förprogrammerade med explicita regler för hur den ska lösa ett problem. Under de senaste åren har stora framsteg inom maskininlärning gjorts. Exempelvis utvecklade DeepMind [3] 2015 en agent som mästrade 49 st Atari-spel [4], med en klassificeringsmodell med endast pixlar och spelpoäng som indata. Under 2016 utvecklade DeepMind sin AlphaGo [5] agent som besegrade en av världens bästa Go spelare, Lee Sedol [6] med 4-1 i matcher. Detta var ett framsteg för AI eftersom Go är ett komplext spel med $2 * 10^{170}$ möjliga drag [7].

Maskininlärning används också för att lösa vardagliga problem. I dagens mobiltelefoner finns en så kallad intelligent personlig assistent, även kallade Siri [8] och Google Assist [9], där användarna får hjälp med t.ex. “vad är det för väder idag?”, “Skicka ett meddelande till mamma att jag blir sen till middagen idag” och “Påminn mig imorgon att jag måste köpa mjölk”. Facebook använder även maskininlärning för ansiktsigenkänning [10] på bilder som laddas upp, och för att rekommendera nya vänner [11].

Figur 2.2 illustrerar att maskininlärning delas upp i två typer av inlärningsprocesser dvs. i övervakad eller oövervakad inlärning och deras underliggande algoritmer.

Oövervakad inlärning används när datamängden bara består av indata och inget förväntat resultat. Målet med oövervakad inlärning är att algoritmen själv lär sig att modellera den komplexa underliggande strukturen så att den kan lära sig mer om datan och själv komma fram till ett resultat [12]. Exempelvis kan en oövervakad klusteralgoritm användas för att hitta likheter i bilder och följaktligen gruppera dem. Övervakad inlärning används när datamängden består av både indata och dess förväntade utdata. Den övervakade algoritmen använder datamängden för att

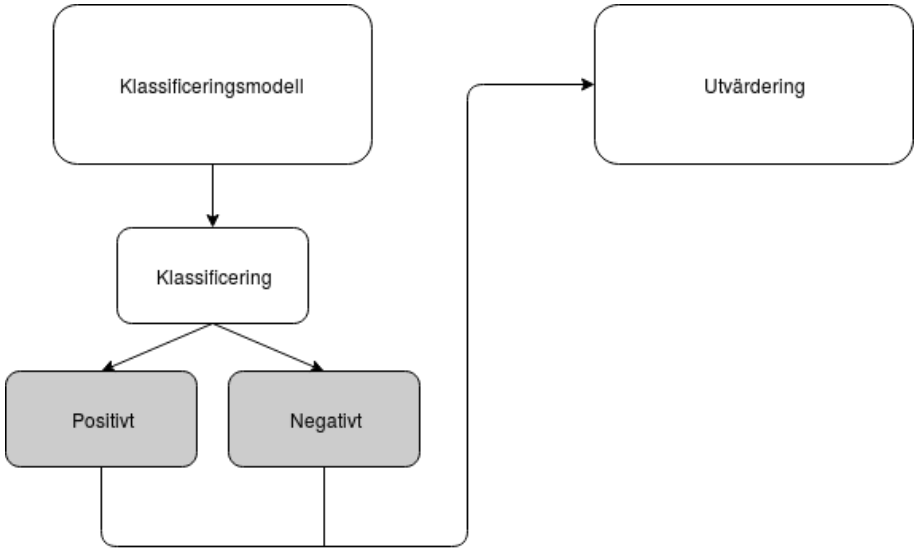


Figur 2.2: Inlärningsprocesser för maskininlärning.

lära sig hur utdata beror på indata genom att skapa en klassificeringsmodell som används för att förutse utdata från ny indata som illustreras i figur 2.3.

Övervakad inlärning kan man tänka sig som att en lärare övervakar programmets inlärningsprocess. Under inlärningsprocessen försöker algoritmen iterativt förutse utdata från datamängden och blir rättad av läraren vid fel förutsägelse [13].

Övervakad inlärning kan brytas ner till klassificerings- och regressionsproblem och eftersom TSA kan kallas ett typiskt klassificeringsproblem [1], kommer studien att använda algoritmer lämpade för klassificeringsproblem. I figur 2.2 kan vi se några av dessa klassificeringsalgoritmer och enligt [2] är speciellt Naive Bayes (NB) och Support Vector Machine (SVM) bäst lämpade för TSA.



Figur 2.3: Klassificeringsprocess av data.

2.2.1 Naive Bayes

NB är en klassificeringsalgoritm som är baserad på Bayes theorem [14] med starka ("naive") oberoende antaganden mellan särdragen d.v.s NB förutsätter att närvaron av ett visst särdrag i en klass inte relaterar till närvaron av ett annat särdrag. Exempelvis kan en frukt anses vara ett äpple om det är grönt, runt och är 10 cm i diameter. Även om särdragen grönt, runt och diameter kan bero på varandra så bidrar alla särdragen självständigt till sannolikheten att frukten är ett äpple, det är därför algoritmen kallas ("naive") [15].

2.2.2 Support Vector Machine

SVM är en övervakad maskininlärningsalgoritm som kan användas till både klassificerings- och regressionsproblem, för det mesta används SVM för klassifikationsproblem. SVM är baserat på idén att hitta ett hyperplan [16] som bäst delar upp datamängden i två klasser [17]. Givet en träningsmängd där förväntad utdata är markerad till en av två kategorier bygger SVM-algoritmen upp en klassificeringsmodell som kan användas för att förutse vilken kategori ny indata ger [18].

2.2.3 Convolutional neural network

Ett av de snabbast växande områden inom maskininlärning är djup inlärning [2]. Begreppet djup inlärning syftar till artificiella neuronnät (ANN) som är uppbyggda med flera lager [19]. ANN är system som försöker ta efter beteende hos biologiska neuronnät för att stegvis bli bättre på att utföra en angiven uppgift genom att studera exempel [20].

CNN är ett av de mest populära neuronnäten [19]. Det var ursprungligen framtaget för maskininlärningsproblem som rör datorseende och har tidigare påvisat imponerande resultat för bildigenkänning [21, 22]. Men på senare tid har det även visat sig att CNN är effektivt att använda för problem som rör sentimentanalys. Med relativt enkla CNN-modeller har imponerande resultat kunnat uppnås inom textklassificering. I [21] presenteras en enkel CNN-modell för textklassificering som påvisade goda resultat över flertalet datamängder. CNN-modellen som har använts i den här studien har utgått från den modell som Yoon Kim presenterar i [21].

2.3 Datamängd

En datamängd för övervakad inlärning består av en mängd in- och utdata som diskuteras i sektion 2.2. Processen för hur rätt utdata (positiv/negativ) märks kan utföras på två olika sätt, antingen genom så kallad mänsklig märkning där människor markerar indata som positiv/negativ eller genom "distant supervision" där en dator märker indata som positiv/negativ utefter någon parameter. Den stora skillnaden mellan mänsklig märkning och "distant supervision" är att "distant supervision" kan generera en mycket större datamängd än mänsklig märkning men risken är större att märkningen blir felaktig [2]. Som är beskrivit i sektion **??** är ett delsytfte med denna studie att utvärdera hur de olika märkningsmodellerna påverkar maskininlärningsalgoritmernas precision. Det finns ett urval med publika twitterdatamängder där både mänsklig märkning och "distant supervison" används [2]. Vi har valt att använda datamängderna Stanford Twitter Sentiment (STS) [23] och SemEval [24] eftersom de är de största publika datamängderna inom respektive märkningsmodell.

2.3.1 SemEval

SemEval-datamängden består av 20633 tweets och är framtagen till en årlig TSA-tävling som har gått sedan 2013 [25]. För datamängden har mänsklig märkning tillämpats, där fem personer manuellt märker varje tweet via Amazon Mechanical Turk [26]. I [27] beskrivs metoden för hur märkningen utförs, varje person markerar varje tweet antingen som mycket positivt, positivt, neutralt, negativt eller mycket negativt. Efter att alla tweets är märkta kartläggs alla tweets till kategorierna positivt, neutralt eller negativt utefter tre kriterier. Antingen att alla personer har märkt en tweet samma, en majoritet har märkt samma eller genom att ta ut ett medelvärde. Medelvärdet räknas ut genom att kartlägga de fem kategorierna till heltal mellan 2 och -2, medelvärdet räknas sedan ut och kartläggs till den närmsta kategorin. För SemEval-datamängden blev utfallet 2760 tweets där alla personer gjort samma märkning, 9944 tweets är där majoritet har märkt samma och för resterande 7928 tweets har medelvärdet räknats ut.

2.3.2 Stanford Twitter Sentiment

STS-datamängden är framtagen mellan April 2009 och Juni 2009 av Alec Go, Richa Bhayani och Lei Huang. STS-datamängden är märkt med hjälp av "distant supervision" där märkningen bestäms av vilken typ av emoji [28] ett twitterinlägg innehåller, exempelvis märks ett twitterinlägg positivt om det innehåller ":), :-),:), :D, =)" och negativt om det innehåller ":(, :-(, : (" [23].

2.4 Twittersentimentanalys

TSA är den del av SA som specifikt handlar om att analysera inlägg som användare gör på Twitter. Twitter är en av de populäraste mikrobloggarna där användare kan skriva och kommunicera med varandra genom twitterinlägg. 2013 var Twitter en av de tio mest besökta sidorna på internet och 2016 uppmättes antalet aktiva användare per månad till 319 miljoner. Twitter är definierat som en mikroblogg på grund av det låga antalet tecken som är tillåtet för ett inlägg. I November 2017

fördubblades antalet tillåtna tecken från de tidigare 140 till 280 [29].

Det finns en rad olika begrepp som kännetecknar Twitter och som är viktiga att känna till [2]. En "tweet" är vad som tidigare nämnts ett twitterinlägg. Det är ett inlägg från en användare som är begränsat till 280 tecken, där användaren exempelvis kan delge sina åsikter i olika ämnen eller dela med sig av personliga upplevelser. En "tweet" behöver inte enbart innehålla ren text utan kan även innehålla länkar, bilder och videor. I fortsättningen av rapporten kommer en "tweet" att benämnas twitterinlägg.

När ett twitterinlägg innehåller "mentions" betyder det att andra användare nämns i inlägget. Det kan vara användbart för att exempelvis delge åsikter om andra användare eller för att öppet starta en diskussion med en nämnd användare. För att nämna en användare i ett twitterinlägg skrivs symbolen @ före användarnamnet.

På Twitter har användare möjligheten att följa andra användare. Det betyder att användare kan följa andra användares aktivitet i deras egna twitterflöden och dela med sig av sin egen aktivitet till sina följares twitterflöden. En användare som följer en annan benämns på Twitter som en "follower". Att följa andra är det primära tillvägagångssättet för att skapa kontakter med andra användare på Twitter.

Användare har möjlighet att kategorisera twitterinlägg och det är vad "hashtags" används för. Genom att använda "hashtags" kan användare märka sina twitterinlägg med etiketter för att knyta inlägget till ett specifikt ämne. Användandet av hashtags gör det enkelt för användare att följa ett ämne. De behöver enbart söka på en specifik "hashtag" för att få fram alla twitterinlägg i ämnet. För att skapa en "hashtag" skrivs symbolen # före namnet på etiketten.

Det är även möjligt att dela andra användares twitterinlägg till ens egna följare. Den funktionen kallas för "retweet" och ett sådant twitterinlägg startar vanligtvis med förkortningen RT följt av en "mention" av den ursprungliga författaren av twitterinlägget. Det kan exempelvis vara användbart för att sprida information till följare eller för att skapa en diskussion om innehållet i twitterinlägget med sina egna följare.

När användare svarar på andras twitterinlägg benämns det som "replies" och det är till för att det ska gå att skapa konversationer, där det ska gå att urskilja vanliga

twitterinlägg från svar på twitterinlägg. En användare svarar på ett twitterinlägg genom att göra en referens till den ursprungliga författaren av inlägget följt av svaret på inlägget.

Användare behöver inte göra alla sina inlägg offentliga för alla användare, de kan begränsa synligheten för deras twitterinlägg att enbart synas för deras egna följare.

I figur 2.4 presenteras ett exempel på hur ett twitterinlägg kan se ut. Twitterinlägget är en "retweet som ursprungligen har skrivits av användaren johanselberg. Det innehåller en "hashtag" med etiketten exempel och en "mention" av användaren KAU. Twitterinlägget innehåller även en extern länk.

Figur 2.4: Twitterinlägg innehållande "hashtag", "mention", "retweet".

2.4.1 Utmaningar

På grund av restriktionen av antalet tillåtna tecken i ett twitterinlägg innehåller majoriteten av twitterinlägg enbart en mening. Därför är det skillnad på klassificeringsnivåerna i TSA och SA. I TSA är det ingen skillnad på dokument- och meningsnivå. Därför används det enbart två klassificeringsnivåer inom TSA: meningsnivå (meddelandenivå) och aspektnivå. [2]

Restriktionen av antalet tillåtna tecken utgör den stora skillnaden mellan TSA och SA. Att analysera sentiment på en text i ett twitterinlägg skiljer sig markant från att göra det på vanliga texter som återfinns i produktrecensioner och nyhetsartiklar. Det gör att TSA ställs inför en rad andra utmaningar än vad SA ställs inför.

I [2] tar författarna upp de viktigaste utmaningar med TSA. För att bra resultat ska uppnås med TSA måste dessa utmaningar hanteras. Det som ligger till grund för utmaningarna med TSA är huvudsakligen restriktionen av antalet tillåtna tecken, att det är en informell typ av medium samt att innehållet på Twitter är dynamiskt och ständigt utvecklas.

Det låga antalet tillåtna tecken och att det är en informell typ av medium, gör att språket som används på Twitter skiljer sig från språket som används i vanlig text. Twitterinlägg innehåller ofta felaktigt språkbruk. Det är vanligt förekommande att Twitterinlägg innehåller förkortningar, slang, nybildade ord och att ord betonas genom att de förlängs eller att de skrivs med versaler.

På grund av att användandet av felaktigt språkbruk är så pass vanligt på Twitter, innehåller twitterinlägg en hel del brus. Felstavade termer gör att antalet gånger en specifik term förekommer i en mängd av text blir mindre. Det resulterar i datagleshet (data sparsity) och har en negativ påverkan på resultatet vid SA. För att minska dataglesheten omvandlas vanligtvis felstavade termer till den korrekta stavningen eller en mer korrekt stavning.

En annan utmaning med TSA är att hantera negationer, vilket även gäller SA. Om negationer förekommer i ett twitterinlägg kan det vända på inläggets sentiment. Därför är det viktigt att kunna tolka och identifiera negationer för att sentimentanalysen ska bli korrekt.

I många fall av SA analyseras texter som är skrivna på ett specifikt språk, exempelvis när nyhetsartiklar utgivna av en viss tidning analyseras. Vid TSA är det inte lika enkelt eftersom twitterinlägg kan vara skrivna på flera olika språk och det är inte ovanligt att språk blandas i inlägg. Den här studien kommer enbart analysera twitterinlägg skrivna på engelska och inhämtade inlägg skrivna på annat språk kommer filtreras bort.

Vid TSA filtreras vanligtvis stoppord bort för att öka prestandan. Stoppord är ord som är vanligt förekommande i texter men som saknar någon större betydelse för texten ifråga. I engelskan är “the”, “is” och “who” exempel på stoppord.

Twitterinlägg behöver inte enbart innehålla text utan de kan även innehålla bilder och videor. Bilder och videor kan ge värdefull information om vad för sentiment som uttrycks i ett twitterinlägg. Det kan exempelvis ge information om vem som uttrycker en åsikt eller om vem en åsikt riktas mot. Den här studien kommer inte att ta hänsyn till mediaobjekt utan kommer enbart att analysera text. Främst på grund av att det i dagsläget är ett outforskat område.

Tabell 2.1: Förvirringsmatris

	Förustspådd positiv	**Förutspådd negativ**
Positiv	Sann positiv	Falsk negativ
Negativ	Falsk positv	Sann negativ

2.4.2 Problem

I [2] listar författarna de problem med TSA som de anser bör utforskas ytterligare. Ett av de viktigaste problemen med TSA anser de vara bristen på datamängder som kan användas som riktmärken vid utvärdering av olika klassificeringsmodeller. Forskning som bedrivs i ämnet använder sig av olika datamängder. Dessutom är det vanligt att forskare själva samlar in och skapar egna datamängder som inte publiceras. Olika datamängder kan generera olika resultat. Därför är det svårt att jämföra olika klassificeringstekniker när det används flertalet olika datamängder. I experimentet, som presenteras i den här studien, har problemet adresserats genom att utvärderingen görs mot två kända och publika datamängder. Delvis för att kunna jämföra hur de olika klassificeringsmodellerna presterar mot olika typer av datamängder, men även för att kunna jämföra resultatet mot andra liknande utvärderingar.

2.5 Utvärdering

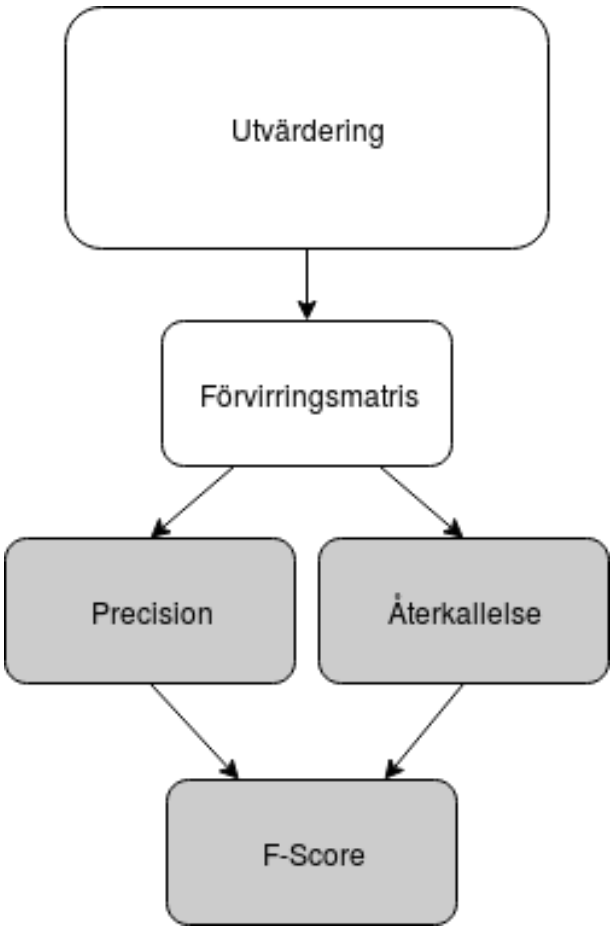
I tabell 2.1 ser vi en så kallad förvirringsmatris (FM) som utvärderar en klassificeringsmodell från datamängden där "positivt" eller "negativt" är förbestämt. Matrisen visar antalet sann positiv (SP), sann negativ (SN), falsk positiv (FP) och falsk negativ(FN) [30]. Med dessa värden kan vi jämföra och analysera modellerna m.h.a följande utvärderingsmetoder: precision (p), återkallelse (å) [31] och F-Score [32].

Precision Är förmågan att modellen inte märker ett tweet som positivt när det är negativt. Detta görs genom att ta antalet sann positiva delat på totalt antal positivt märkta tweets.

Återkallelse Är förmågan att modellen märker positiva tweets korrekt. Detta görs genom att ta antalet sann positiva delat på summan av antalet sann positiva och falsk negativa.

F-Score Även kallat det harmoniska medelvärdet mellan precision och återkallelse används då inte alltid precision och återkallelse räcker till för att göra en helhetsbedömning.

I figur 2.5 illustreras hur utvärderingmetoderna hänger ihop.



Figur 2.5: Utvärderingsmetoder för TSA.

2.6 Sammanfattning

I detta kapitel har bakgrunden till projektet diskuterats och där nyttan av SA och TSA har tagits upp. Intressant för uppdragsgivaren är hur de kan integrera SA och TSA i sina produkter t.ex. chatbotar och trendanalys.

Detta kapitel ger även en överblick av vad maskininlärning är och hur maskininlärningens viktigaste komponenter hänger ihop samt hur maskininlärning kan appliceras på SA och TSA. Dessutom beskrivs de problem och utmaningar som existerar inom TSA.

Kapitel 3

Experiment

I kapitlet kommer experimentet för studien att presenteras, hur det har utförts och hur de olika delarna har implementeras. Delarna utgörs av bearbetning av datamängderna, vilka särdragsurval som kommer utföras och en fördjupning i hur maskininlärningsalgoritmerna fungerar. Implementationen kommer ske i tre steg där först en lexikonbaserad modell kommer tas fram, för att ge ett basfall för respektive datamängd, för att sedan implementera algoritmerna med standardparametrar och slutligen testa och justera algoritmernas parametrar. Detta för att eventuellt uppnå en högre precision för respektive klassificeringsmodell. De slutgiltliga klassificeringsmodellerna kommer att diskuteras och jämföras i avsnitt 4.

3.1 Bearbetning av datamängd

En viktig del inom SA är bearbetningen av den data som ska analyseras. Bearbetning av data, i fallet SA, handlar om att tvätta och förbereda texter som ska klassificeras [33].

Kvaliteten på data som ska analyseras är avgörande för vilka resultat som kan uppnås vid maskininlärningen [33]. I [34] rapporteras att bearbetning av text innan maskininlärning kan ha en tydlig positiv påverkan på en klassificeringsmodells prestanda vid sentimentanalays. Därför behöver data som ska analyseras bearbe-

tas innan maskininlärningen, vilket betyder att data normaliseras och reduceras på brus. Teorin med att bearbeta en datamängd innan maskininlärningen är att det delvis kan förbättra prestandan för klassificeringsmodellen men även att klassificering kan utföras snabbare, vilket kan vara av betydelse för sentimentanalys som utförs i realtid [35].

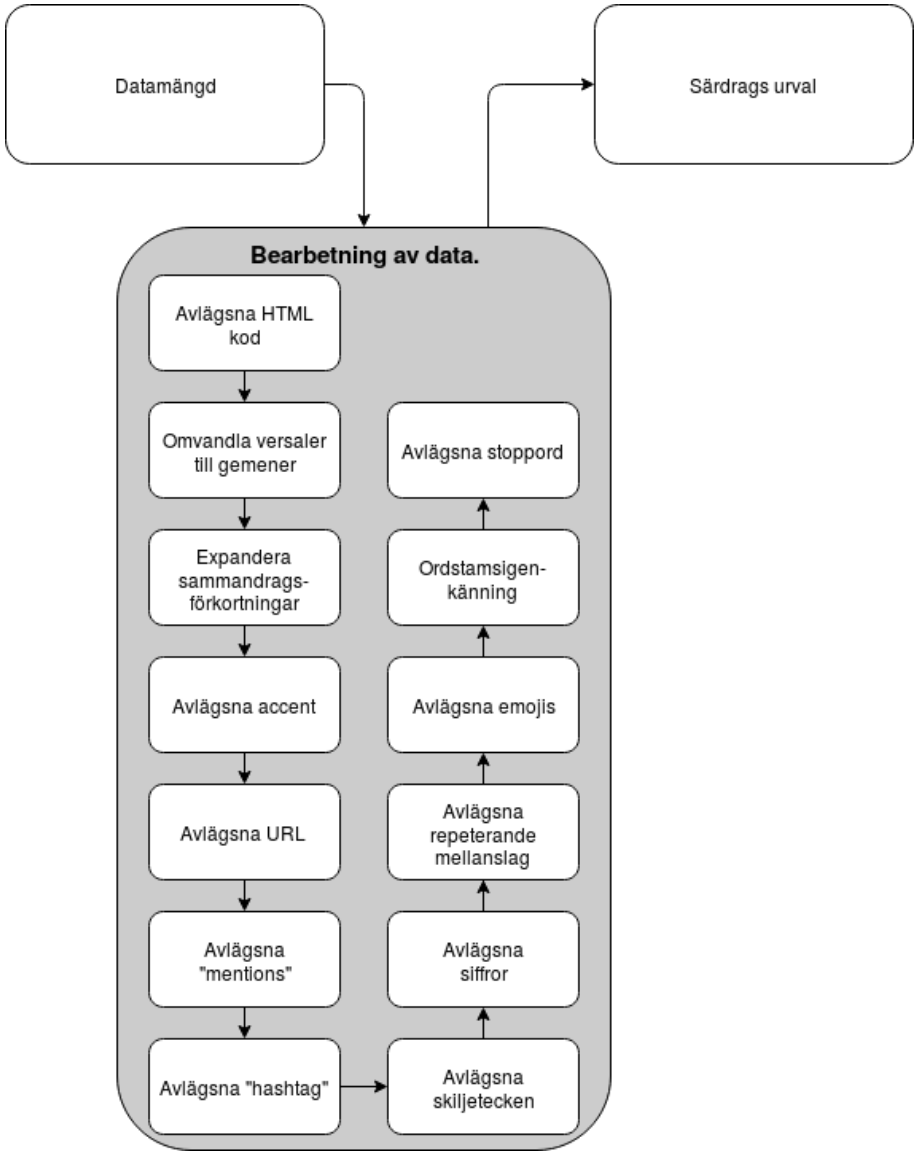
Som nämndes i avsnitt 2.4.1, innehåller twitterinlägg vanligtvis stora mängder brus på grund av det informella språket som används på Twitter, dvs data som inte är användbar för analysprocessen. Det finns exempelvis många ord som inte har någon påverkan på en texts sentiment och därför är det vanligtvis inte nödvändigt att ta med alla ord från den ursprungliga texten vid maskininlärning [33, 34]. Ord som dessa kan tas bort, ersättas eller slås samman med andra. Det är det som kallas för att bearbeta data. Genom att göra det minskas dimensionen på problemet och textklassificeringen blir enklare att utföra , eftersom varje ord behandlas som en dimension [34].

Det existerar flertalet olika tekniker som kan användas vid bearbetningen av data, vissa mer vanliga än andra. I arbetet med studien har två artiklar använts som undersöker effekten av olika bearbetningstekniker vid TSA, [33] och [35]. Båda artiklarna utvärderar bearbetningsteknikerna utefter hur de påverkar prestandan vid sentimentklassificering.

I [33] utvärderas 15 olika bearbetningstekniker. Teknikerna utvärderas var för sig och testas mot två olika datamängder, där de för varje datamängd testas med tre olika klassificeringsmodeller. Några bearbetningstekniker ger bättre noggrannhet för båda datamängderna, andra sämre och resultaten varierar för några. Resultaten varierar inte enbart mellan datamängderna utan de varierar även mellan klassificeringsmodellerna. De tekniker som rekommenderas av författarna och resulterar i hög noggrannhet för alla klassificeringsmodeller och de båda datamängderna är: borttagning av nummer, omvandling av ord till deras ursprungliga form samt ersättning av upprepade skiljetecken. Hantering av negationer, lemmatisering samt ersättning av URL:er och “mentions” av användare påvisar också bra resultat för båda datamängderna men dock inte för alla klassificeringsmodeller. Borttagning av skiljetecken var den bearbetningsteknik som gav sämst resultat. Det som saknas i [33] är hur de olika förbehandlingsteknikerna presterar i kombination med varandra, vilket enligt författarna själva kan ge andra resultat.

Detta är dock något som görs i [35]. Istället för att utvärdera teknikerna isolerade från varandra som i [33], utvärderas de istället utifrån hur de presterar i kombination med andra. Dock har de enbart valt att fokusera på sex olika bearbetningstekniker: expandering av sammandragsförkortningar, expandering av förkortningar, avlägsning av nummer, avlägsning av stoppord, ersättning av förlängda ord och avlägsning av URL:er. Bearbetningsteknikerna har utvärderats utefter hur de presterar på fem olika datamängder på fyra klassificeringsmodeller. Vid bedömningen av prestandan vid sentimentklassificeringen har ett basfall använts. Basfallet har använt samtliga sex bearbetningstekniker för bearbetningen av datan vid testningen. När en specifik teknik har utvärderats, har tekniken i fråga exkluderats från mängden av bearbetningstekniker. Därefter har testerna repeterats, med fem bearbetningstekniker istället för sex. Sedan har förändringen av resultatet gentemot basfallet använts för att bedöma hur tekniken påverkar prestandan vid sentimentklassificeringen. Resultaten visar att expandering av sammandragsförkortningar och expandering av förkortningar kan ha positiv påverkan på noggrannheten vid klassificeringen. Avlägsning av URL:er, nummer och stoppord påverkar resultaten för klassificeringen minimalt, men är effektiva för att minska brus.

Utifrån resultaten som har presenterats i [33] och [35] har bearbetningsteknikerna för experimentet i den här studien valts. Det svåra med att välja bearbetningstekniker, utifrån litteraturen, är att det kan existera flera olika varianter av en viss teknik och att olika namn kan användas för att beskriva en viss teknik. I [33] hanterar de exempelvis URL:er genom att ersätta varje URL med etiketten “URL”, medan de avlägsnas helt i [35]. Eftersom bearbetningsteknikerna uteslutande kommer att användas i kombination med varandra vid det här experimentet har störst vikt lagts på de tekniker som presenteras i [35].



Figur 3.1: Bearbetningstekniker som används för att bearbeta data vid experimentet.

Nedan presenteras och beskrivs de bearbetningstekniker som har valts att användas för experimentet. De valda teknikerna visualiseras i figur 3.1. För några av bearbetningsteknikerna har ordningen betydelse. Exempelvis kommer inte expanderingen av sammandragsförkortningar kunna genomföras ifall avlägsnandet av accenter sker innan. Totalt har 13 bearbetningstekniker använts. Majoriteten

av dessa handlar om att avlägsna text från twitterinläggen. Det som avlägsnas vid bearbetningen av twitterinläggen är: HTML-kod, accenter, URL:er, “mentions”, “hashtags”, skiljetecken, siffror, repeterande mellanslag, emojis och stoppord. Avlägsning av emojis diskuteras i avsnitt 3.2. Stoppord, som beskrevs i 2.4.1, tas bort genom att twitterinläggen söks igenom på stoppord från en fördefinierad lista. Trots att man i [33] rapporterar sämre prestanda vid avlägsning av skiljetecken har det valts att användas som en bearbetningsteknik för det här experimentet. Det valet gjordes eftersom att man i [33] samtidigt rapporterar att det bidrar till att reducera storleken på klassificeringsproblem.

De övriga teknikerna som används omvandlar text under bearbetningen av ett twitterinlägg. Ett av de första stegen i den bearbetningsprocess, som har använts för detta experiment, är att omvandla versaler i ett twitterinlägg till gemener. Det gör att ord som är skrivna på flera olika sätt slås samman till ett [33]. Vokabulär blir då mindre, vilket gör att storleken på problemet minskas.

En annan teknik som används är att expandera sammandragsförkortningar. I 2.4.1 diskuterades vikten av att kunna tolka och identifiera negationer för att SA ska bli korrekt. Eftersom sammandragsförkortningar är vanligt förekommande i engelskan fokuserar tekniken på att expandera sammandragsförkortningar med motsägelser. Exempelvis är termen “don’t” exempel på en sammandragsförkortning för motsägelsen “do not”. För experimentet har sammandragsförkortningarna “n’t”, “can’t” och “won’t” omvandlats till “not”, “can not” och “will not”, utefter hur metoden beskrevs i [35].

Slutligen har också ordstamsigenkänning använts vid bearbetningen av ett twitterinlägg. Ordstamsigenkänning går ut på att ta bort ändelser från ord för att omvandla dem till deras ursprungliga form [33]. Det leder också till att ord slås samman och att storleken på problemet minskar. Tabell 3.1 visar ett exempel på vad som sker då ordstamsigenkänning används på olika böjningar av det engelska ordet ”arrive”.

Tabell 3.1: Exempel på vad som sker när ordstamsigenkänning används på olika böjningar av det engelska ordet ”arrive”.

Före	**Efter**
Arriving	Arriv
Arrive	Arriv
Arrives	Arriv

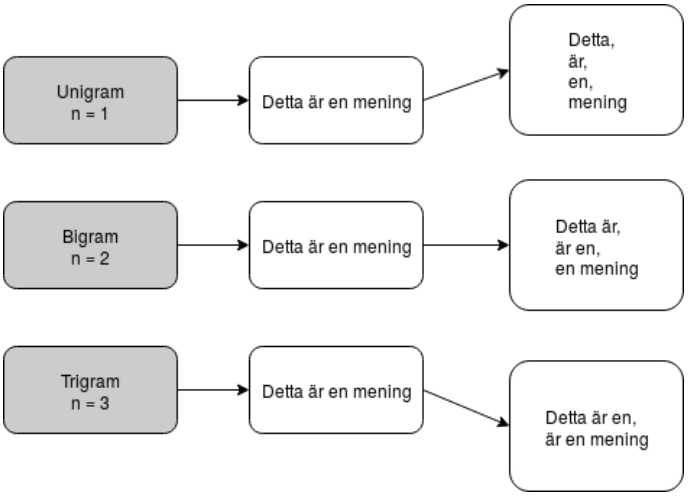
Tabell 3.2 visar ett exempel på vad som sker när ett twitterinlägg bearbetas med bearbetningsteknikerna som är valda för experimentet.

Tabell 3.2: Exempel på vad som sker med ett twitterinlägg efter det har bearbetas med alla valda bearbetningstekniker.

Före	@SportsCenter Hands down the Irish amateur, Paul Dunne, thrilled the world. More to come tomorrow!!! #NICE https://t.co/kTYNkltfl6
Efter	hand irish amateur paul dunn thrill world come tomorrow nice

3.2 Särdragsurval

Inom maskininlärning definieras ett särdrag som en individuell mätbar egenskap eller kännetecken på ett fenomen som har observerats, begreppet särdrag kommer från variabler som används inom statistik. Att välja korrekta, informativa och oberoende särdrag är ett kritiskt moment för att maskininlärningsalgoritmen ska kunna uppnå sin fulla potential och undvika fel klassificeringar [36]. I [2] presenteras det fyra olika särdragsklasser inom TSA: semantiska, syntaktiska, stilistiska och twitter-specifika särdrag.



Figur 3.2: Illustration av n-grams på en mening.

Semantiska särdrag används mestadels för att ta ut åsiktsord, sentimentord och negationer. Åsiktsord är ord eller meningar som kan innehålla någon typ av åsikt medan sentimentord är ord som innehåller något positivt eller negativt. Negationer är ett viktigt koncept eftersom en mening som innehåller någon typ av negation kan skifta om twitterinlägget är positivt eller negativt. Studien kommer använda sentimentord i lexikonimplementationen som diskuteras i avsnitt 3.4.2 och negationer som även diskuterats i avsnitt 3.1 kommer implementeras i avsnitt 3.4.1.

Syntaktiska särdrag används för att utforska påverkan av olika termer i SA och TSA, syntaktiska särdrag bryts ner till unigram, bigram, trigrams, n-gram, termfrekvens, omvänd dokumentfrekvens (IDF) och "Part of Speech" (POS). Där n-grams är ett samlingsord för uni-, bi- och trigram, i figur 3.2 illustreras hur n-gram modellen fungerar på en enkel mening och hur meningens ord grupperas på olika sätta till ett ord eller fras.

Termfrekvens används då längre twitterinlägg kan ha ett högre antal förekomster av ett ord än ett kortare twitterinlägg, vilket kan leda till att ord kan få större betydelse för klassificeringen. Genom användandet av termfrekvens delas antalet förekomster av ett ord i ett twitterinlägg med alla ord som finns i twitterinlägget undviks sådana problem [37]. IDF [38] används i kombination med termfrekvens eftersom ordet "the" är så pass vanligt, kommer termfrekvens att lägga för stor vikt

på ordet “the” som egentligen inte säger något om vilket sentiment twitterinlägget har. Så m.h.a IDF kommer ord som används sällan få större vikt och eventuell högre precision kan uppnås.

POS utförs genom att man räknar hur många substantiv, verb och adjektiv som existerar i ett twitterinlägg. POS kan t.ex användas för att ta reda på åsikter i ett twitterinlägg där antalet adjektiv kan relateras till vilken åsikt twitterinlägget har.

Studien kommer implementera unigram som ett bassärdrag för NB och SVM för att sedan utforska n-grams och termfrekvens för eventuell förhöjd prestanda vilket diskuteras i avsnitt 3.4. POS kommer inte implementeras i studien då [39, 40] rapporterar försämrad precision. Däremot rapporteras i [41] små förbättringar vid användning av POS.

Stilistiska särdrag är särdrag som kommer från det informella skrivsättet som används på Twitter där t.ex emojis, förkortningar, slang och skiljetecken används. Enligt [41] kan emojis ha en stor betydelse för TSA. Men eftersom STS-datamängden har avlägsnat emojis, då “distant supervision” [23] har använts, kommer inte studiens modeller ta hänsyn till emojis. Därför har även emojis från SemEval-datamängden filtrerats bort i avsnitt 3.1. Studien kommer inte hantera slang eftersom inget större slanglexikon var öppet för användning. Förkortningar och skiljetecken kommer hanteras i denna studie och har diskuterats i avsnitt 3.1 och implementeras i avsnitt 3.4.1.

Twitter-specifika särdrag är särdrag som är specifika för Twitters domän som t.ex “hashtags” och “mentions” vilka diskuterats i 2.4 och 3.1.

3.3 Algoritmer

För att ge en bättre förståelse för hur NB, SVM och CNN kan appliceras på sentimentanalys, kommer detta avsnitt ge en överblick av hur algoritmerna går till väga för att klassificera text och hur de kan anpassas för att uppnå bra resultat vid sentimentanalys.

3.3.1 Naive Bayes

Det finns tre varianter utav NB nämligen Gaussian, Bernoulli och Multinomial. Gaussian används när attributen är kontinuerligt fördelade och då antas det att attributen förknippade med varje klass fördelas enligt Gaussian d.v.s normalfördelning [42]. Bernoulli användas när attributen är fördelade enligt multivariat Bernoulli [43] och Multinomial NB (MNB) används när attributen är multinomial fördelat. I [44] diskuteras det att MNB är väl lämpad för text klassificering vilket gör att studien kommer använda MNB.

I 2.2.1 beskrivs att Naive Bayes-algoritmen är baserad på Bayes theorem [14] vilket för TSA betyder att räkna ut sannolikheten att ett twitterinlägg (B) är positivt eller negativt (A) för en mängd C, bestående av ett antal B med ett förbestämt A. Där $P(A|B)$ är en villkorlig sannolikhet för att A inträffar givet att B är sant, $P(B|A)$ är också en villkorlig sannolikhet för att B inträffar givet att A är sant. $P(A)$ och $P(B)$ är sannolikheten att observera $P(A)$ och $P(B)$ oberoende av varandra,

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}. \tag{3.1}$$

För att ekvation 3.1 skall vara "Naive" antas att varje ord b_w i B är oberoende av varandra vilket betyder att vi inte längre klassificerar B utan b_w, $P(B|A) = P(b_{w1}|A) * P(b_{w2}|A) * \ldots * P(b_{wn}|A)$, där n är antalet ord i B,

För att beräkna $P(b_w|A)$ används Dirichlet prior [45, 46],

$$P(b_w|A) = \frac{N_{b_w} + \alpha}{N_A + \alpha_n}. \tag{3.2}$$

Där N_{b_w} är hur många gånger b_w förekommer i delmängden C_A, N_A är summan av alla ord i C_A, α och α_n används för att utföra additiv utjämning [47] där $\alpha = 1$ och α_n är summan av antal olika ord i C.

Tabell 3.3: Exempeldatamängd med twitterinlägg som är märkta som positiva eller negativa.

Twitterinlägg	**Märkning**
ser fram emot en dag i solen	Positiv
jävla skitväder	Negativ
vilken jävla kung du är linkan	Positv
fyfan vilken dålig match	Negativ

I tabell 3.3 illustreras en datamängd C med fyra twitterinlägg och deras märkning. Denna datamängd kan användas för att beräkna om twitterinlägget "solen skiner alltid i karlstad" (B) är positivt eller negativt (A) . Genom att räkna ut sannolikheten $P(Positiv|B)$ och $P(Negativ|B)$ kan märkning bestämmas, detta görs genom att jämföra vilken sannolikhet som blir störst och vi får följande ekvation,

$$P(A|B) = \prod_{i \epsilon B} \frac{P(i|A)}{P(B)}, \tag{3.3}$$

Och eftersom nämnaren är lika i bägge uträkningarna kommer enbart täljaren vara relevant vid beräkning.

För att räkna ut exempelvis $P(solen|Positiv)$ används ekvation 3.2 där $N_{b_w} = 1$, $N_A = 13$, $\alpha_n = 17$ och $\alpha = 1$ vilket ger oss $P(solen|Positiv) = \frac{1+1}{13+17} = 0,0667$. I tabell 3.4 illustreras alla ekvationer för att beräkna om "solen skiner alltid i karlstad" är positiv eller negativ. Ovanstående utförs på varje ord för både positiv och negativ vilket illustreras i tabell 3.4. Med hjälp av tabell 3.4 och ekvation 3.3 beräknas,

$$P(Positiv|\text{solen skiner alltid i karlstad}) = \frac{1.6428 * 10^-7}{P(B)}, \tag{3.4}$$

och

$$P(Negativ|\text{solen skiner alltid i karlstad}) = \frac{1.5576 * 10^-7}{P(B)}. \tag{3.5}$$

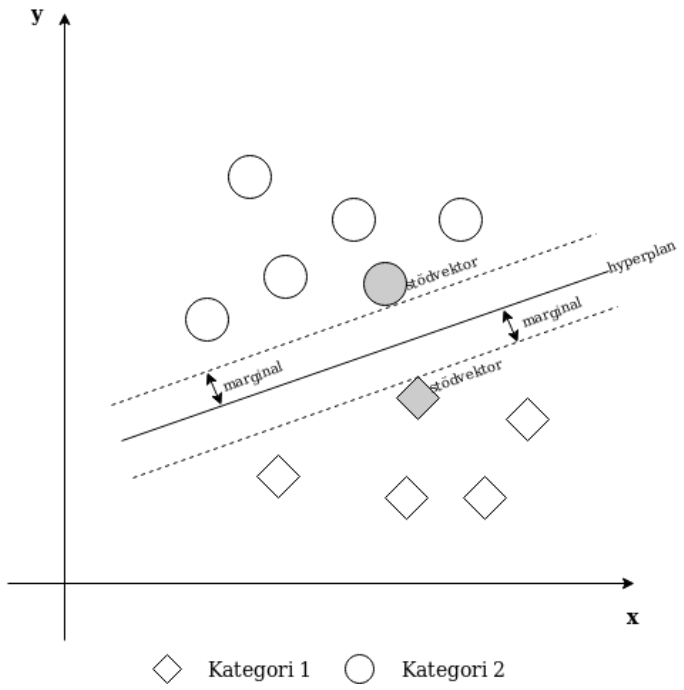
Jämförs dessa resultat kommer twitterinlägget klassificeras som positivt.

Tabell 3.4: Beräkningar för om ett ord är positivt eller negativt.

Ord	**P(ord|Positiv)**	**P(ord|Negativ)**
solen	$\frac{1+1}{13+17}$	$\frac{0+1}{6+17}$
skiner	$\frac{0+1}{13+17}$	$\frac{0+1}{6+17}$
alltid	$\frac{0+1}{13+17}$	$\frac{0+1}{6+17}$
i	$\frac{1+1}{13+17}$	$\frac{0+1}{6+17}$
karlstad	$\frac{0+1}{13+17}$	$\frac{0+1}{6+17}$

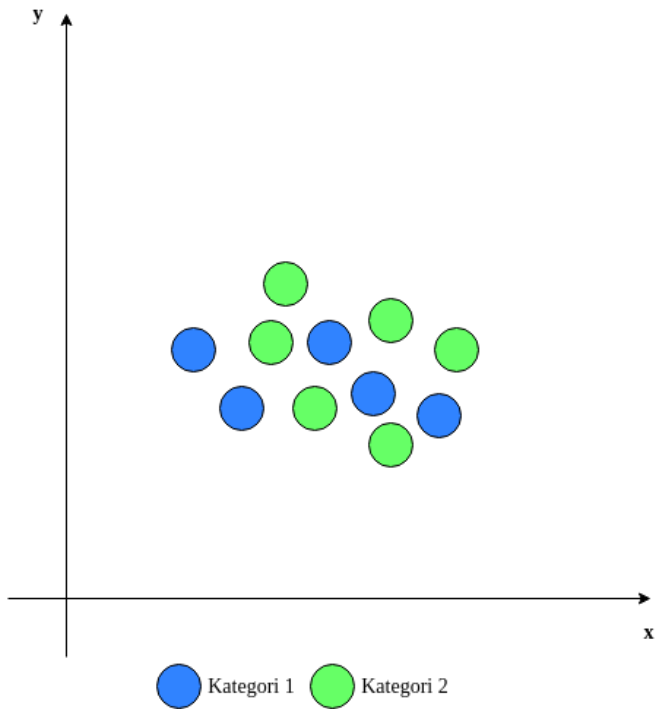
3.3.2 Support Vector Machine

I avsnitt 2.2.2 beskrevs SVM kortfattat, i detta avsnitt kommer SVM att beskrivas mer ingående. SVM-algoritmen skapar en modell utifrån en given mängd märkt träningsdata [18]. Modellen som skapas har sedan förmågan att kategorisera ny indata i en av de två kategorierna. Den skapade SVM-modellen representerar träningsdata som punkter i ett koordinatsystem. I koordinatsystemet kartläggs data så att de två kategorierna är tydligt separerade med ett så stort mellanrum som möjligt. När modellen kategoriserar ny indata kartläggs den datan till samma koordinatsystem, för att sedan kategoriseras utefter vilken sida av mellanrummet den hamnar på.



Figur 3.3: Exempel på en linjär klassificerare där de två kateogrierna är linjärt separabla.

SVM separerar kategorierna genom att konstruera ett hyperplan [17]. Datapunkterna som hyperplanet positioneras utefter kallas stödvektorer. De är datapunkterna som ligger närmast hyperplanet från respektive kategori som ska klassificeras. I figur 3.3 representeras hyperplanet av linjen mellan kategorierna. Det kan finnas flera möjliga hyperplan som kan kategorisera datan. Idén bakom SVM är att hitta hyperplanet som bäst delar upp datamängden i två kategorier. Ju större marginal mellan kategorierna desto mindre känslig blir modellen för generaliseringsfel [18]. Därför anses hyperplanet som har störst marginal till stödvektorerna från respektive kategori ge en bra separation av data.

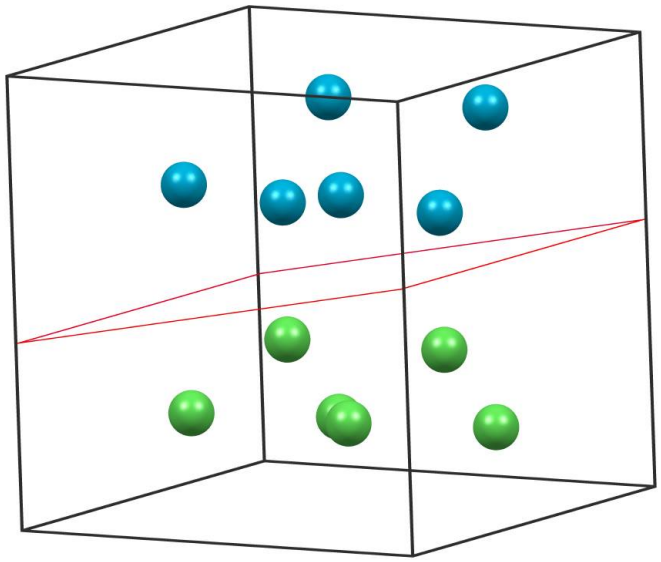


Figur 3.4: Exempel på en icke-linjär klassificeringsuppgift, där de två kategorierna inte är linjärt separabla.

Datapunkterna som SVM hanterar är vektorer av p dimensioner, där varje dimension representerar ett särdrag. [18]. En linjär klassificerare definieras som en klassificerare där datapunkterna från de två kategorierna kan separeras med ett hyperplan av $p - 1$ dimensioner. Klassificeringsuppgiften i figur 3.3 har två särdrag: x och y [48]. Datapunkterna i exemplet representeras således av en vektor av 2 dimensioner och de två kategorierna kan separeras med en linje. Exemplet i figur 3.3 representerar en linjär klassificerare, då de två kategorierna är linjärt separabla.

Dock finns det klassificeringsproblem som inte går att separera linjärt, så kallade icke-linjära klassificeringsproblem. Figur 3.4 visar ett exempel på ett sådant problem. SVM kan även hantera sådana klassificeringsproblem [18]. I fall som dessa ritar SVM om datapunkterna till en högre dimension för att en separation ska bli möjlig, eftersom separationen förmodligen blir enklare i en högre dimension. Processen upprepas tills dess att ett hyperplan kan separera datan. Metoden SVM

använder för att utföra detta effektivt kallas “kernel trick” [49]. Detta illustreras i figur 3.5, där exemplet från figur 3.4 har ritats om i en högre dimension. Efter omritning till en högre dimension kan kategorierna separeras med ett plan.



Figur 3.5: Exemplet från figur 3.4 ritas om till en högre dimension.

Om ett problem kräver en linjär eller en icke-linjär klassificerare beror på hur den data som ska klassificeras ser ut. Beteendet för SVM-algoritmen går att anpassa utefter den data som ska klassificeras genom att välja olika “kernel”-funktioner [50]. För textklassificering är det rekommenderat att använda en linjär “kernel”, vilket ger en linjär klassificerare. Anledningen till det är dels att de flesta textklassificeringsproblem är linjärt separerbara. Men det är även bra att använda en linjär “kernel” när antalet särdrag för klassificeringsproblemet är högt, vilket är fallet vid textklassificering. När antalet särdrag är högt är det ingen märkbar skillnad i prestanda mellan en linjär kontra icke-linjär “kernel”. Dessutom går en SVM-modell med en linjär “kernel” snabbare att träna än andra och det är färre parametrar som behöver justeras vid optimering av en modell. På grund av ovanstående anledningar har en linjär klassificerare valts att användas vid den här studien.

3.3.3 Convolutional Neural Network

I avsnitt 2.2.3 beskrivs kortfattat vad CNN är och varför det på senare tid har börjat användas för maskininlärningsproblem som rör textbearbetning. I detta avsnitt kommer CNN att beskrivas på en mer teknisk nivå. De olika delarna som ett CNN består av kommer att presenteras, och därefter beskrivs hur ett CNN kan utföra textklassificering utifrån ett exempel.

CNN är uppbyggt med flera lager [51]. Det har ett lager för indata, ett för utdata och dessutom ett flertal dolda lager. De dolda lagrena består av konvulsionlager, "pooling"-lager, fullt anslutna lager och normaliseringslager.

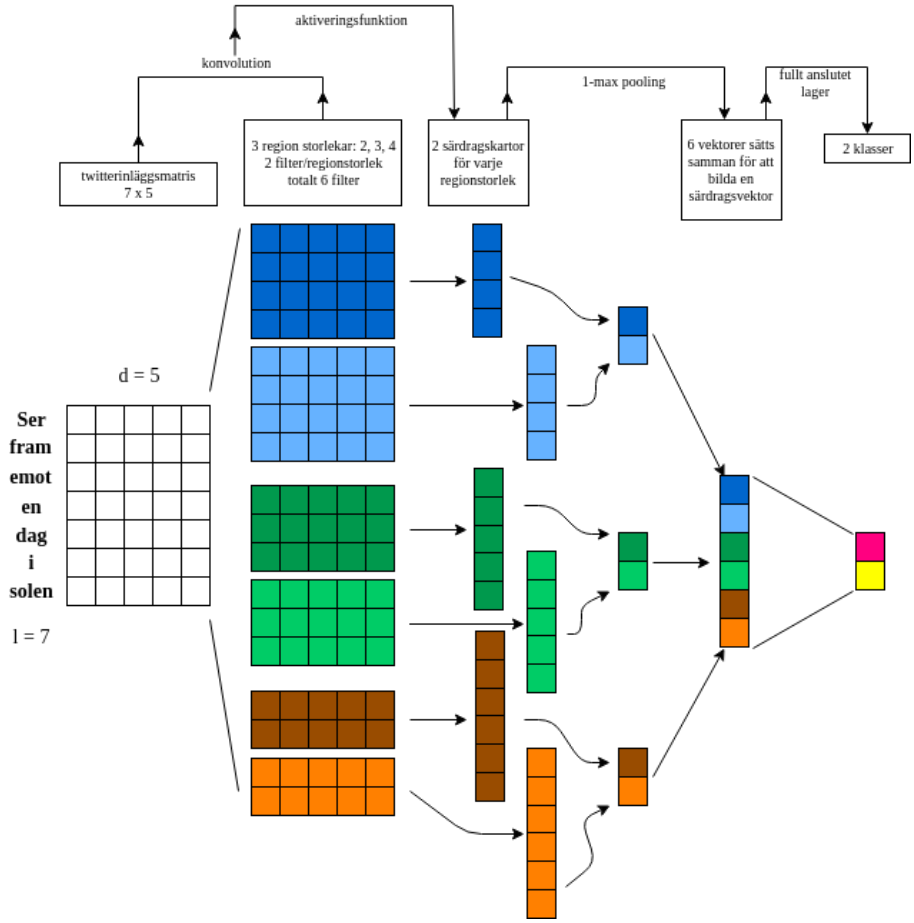
Konvulsionlager utgör den centrala delen av CNN [51]. Ett konvulsionlager består av ett antal lärbara filter. I [52] liknas en konvulsion med en funktion där ett fönster förflyttas över den data som ska analyseras, där datan är representerad som en matris. Fönstret som förflyttar sig över matrisen är det som kallas för filter. Vid varje förflyttning beräknas summan efter att filtrets värden har multiplicerats med respektive värde i matrisen. En hel konvulsion ges efter att fönstret har bearbetat hela matrisen.

"Pooling"-lager används vanligtvis efter konvultionslager [52]. Det "pooling"-lager gör är att de tar ut en delmängd av den data de får in. Detta kan utföras med olika metoder, men en vanlig metod är att ta ut det största värdet från varje filter som bearbetas. Denna strategi kallas för "1-max pooling" [53]. För klassificering krävs vanligtvis att de matriser som ges av filtren har samma storlek [52]. Genom användandet av "pooling" uppfylls det kravet, t.e.x om 100 filter används och "1-max pooling" appliceras på varje lager, kommer det alltid resultera i en vektor på 100 dimensioner. "Pooling" gör det alltså möjligt att skicka in data av varierande storlek och använda olika filterstorlekar, och alltid få ut den data som ska tolkas vid klassificeringen i samma storlek. En annan anledning till att "pooling"-lager används är att det minskar storleken på den data som ska analyseras samtidigt som den försöker behålla den mest relevanta informationen.

Efter att olika särdrag har tagits fram genom konvulsionlager och "pooling"-lager kopplas ett fullt anslutet lager in i slutet av nätverket [54]. Det fullt anslutna lagret tar in mängden av särdrag från lagret innan och skickar ut en vektor av storleken N, där N representerar antalet klasser vid klassificeringen. Värdena i

denna vektor representerar sannolikheten för respektive klass. Det fullt anslutna lagret analyserar den data som skickas från lagret innan och gör en bedömning utefter vilken klass särdragen med störst sannolikhet representerar.

Som nämndes i 2.2.3, är CNN ursprungligen framtagen för problem som rör bearbetning av bilder. När bilder bearbetas av CNN representeras bildens pixlar i en matris, som filtrena sedan förflyttar sig över [52]. Text som ska bearbetas av CNN behöver därför också representeras som en bild. I figur 3.6 visas ett exempel av ett CNN för textklassificering. I exemplet ska ett twitterinlägg av längden l klassificeras [53]. Twitterinlägget som ska klassificeras har konverterats till en twitterinläggmatris. Varje ord i twitterinlägget utgör en rad i matrisen och representeras av en vektor i matrisen, där vektorernas dimension d utgör matrisen bredd. Matrisens storlek ges således av $l*d$. Vektorerna som representerar olika ord är antingen ordinbäddningar [55] eller "one-hot"-vektorer [52]. "One-hot"-vektorer är det som har använts för implementationen av CNN i denna studie och där indexeras varje ord i datamängden till ett vokabulär. Efter att twitterinlägget har konverterats till att representeras som en matris, går det att hantera inlägget som en bild och utföra konvulsioner på det genom olika filter [53]. Eftersom varje rad representerar ett ord i twitterinlägget är det rimligt att använda filter som till bredden är lika stora som dimensionen på vektorerna. Höjden eller regionstorleken på filtren kan däremot varieras och det går att använda flertalet filter med olika regionstorlekar. Regionstorleken anpassas utefter hur många ord ett filter ska omfatta.



Figur 3.6: Exempel på ett CNN som används för textklassificering. Figur inspirerad av SÄTT IN MANUELLT CNN:002.

I exemplet som finns i figur 3.6 används tre olika regionstorlekar och två filter för varje regionstorlek, totalt används sex filter. Genom användandet av flertalet filter för respektive regionstorlek går det att få ut komplementerande särdrag från samma region [53]. Från konvulsionerna som utförs på matrisen genereras särdragskartor. Den engelska termen är “feature maps”, men i denna studie kallas de för särdragskartor. Storleken på varje särdragskarta som genereras kommer att variera beroende på twitterinläggets längd och regionstorleken på respektive filter. För att omvandla vektorerna till samma längd används en “pooling”-funktion på varje särdragskarta. Dessa vektorer sätts sedan samman till en ny vektor där twitterinläggets särdrag är representerade på en högre nivå. I slutet av nätverket har

ett fullt anslutet lager kopplats in. Det fullt anslutna lagret tar slutligen in den sammankopplade vektorn och utför klassificeringen.

3.4 Implementation

Detta kapitel beskriver detaljer angående implementationen av vår studie och dess flöde som illustreras i figur 2.1. Kapitlet beskriver även teknikerna som används för att göra implementationen möjlig. Gemensamma tekniker som används i alla delar av implementation är programmeringsspråket Python3 [56] och Pandas [57]. Pandas är ett Python-paket som erbjuder snabba, flexibla datastrukturer som är utformade för att göra arbetet med relationell och märkt data smidigare.

3.4.1 Databearbetning

Som beskrivs i avsnitt 3.1 är databearbetning ett viktigt steg inom SA och TSA. I detta avsnitt beskrivs implementationen av databearbetningen m.h.a delarna som illustreras i figur 3.1.

För att avlägsna HTML-kod från twitterinlägg används BeautifulSoup [58], som är ett Python-paket för att extrahera data från HTML-kod, om HTML-kod finns i twitterinlägg tas det således bort, se kodavsnitt 3.1.

Kodavsnitt 3.1: Funktion för att avlägsna HTML-kod.

```
def remove_html_encode(self):
       self.df["tweet"] = self.df["tweet"]./
       apply(lambda x: BeautifulSoup(x, 'lxml').get_text())
```

Omvandling från versaler till gemener utförs på varje twitterinlägg m.h.a den inbyggda Python-funktionen `str.lower()` som konverterar versaler till gemener, se kodavsnitt 3.2.

Kodavsnitt 3.2: Funktion för att omvandla versaler till gemener.

```
def to_lower(self):
      self.df["tweet"] = self.df["tweet"].str.lower()
```

Beskrivet i 3.1 är expandering av sammandragsförkortningar ett viktigt moment för att kunna hantera negationer. Med `cont_dic` kan vi hantera och expandera “can’t”, “won’t” och andra ord som slutar på “n’t”, exempelvis expanderas "isn’ttill "is not". Se kodavsnitt 3.3

Kodavsnitt 3.3: Kod för att expandera sammandragsförkortningar.

```
cont_dict = {'can\'t': 'cannot', 'won\'t': 'will not', 'n\'t': ' not'}
cont_re = re.compile('(%s)' % '|'.join(cont_dict.keys()))

def expand_cont(self, s, cont_dict=cont_dict):
       def replace(match):
                  return cont_dict[match.group(0)]
        return cont_re.sub(replace, s)

def expand_contractions(self):
       self.df["tweet"] = self.df["tweet"].apply(lambda x: self.expand_cont(x))
```

Avlägsna accenter utförs genom att använda `unicodedata.normalize()` [59] som använder “Normalization Form Compatibility Decomposition” (NFKD) parametern, vilket betyder att `unicodedata.normalize()` byter ut en bokstav med accent mot sin ASCII-ekvivalenta bokstav. Resterande parametrar säger att om bokstaven redan är ASCII eller utf-8 ignoreras dessa bokstäver, se kodavsnitt 3.4.

Kodavsnitt 3.4: Funktion för att avlägsna accenter.

```
def remove_accented_chars(self):
    self.df["tweet"] = self.df["tweet"].apply(lambda x:unicodedata./
    normalize('NFKD', x).encode('ascii', 'ignore')./
    decode('utf-8', 'ignore'))
```

I kodavsnitt 3.5 avlägsnas URL:er, “mentions”, “hashtags” , skiljetecken, siffror, emojis och repeterande mellanslag m.h.a fyra reguljära uttryck (RU) [60]. För att avlägsna URL:er används `'https?://[A-Za-z0-9./]+'` vilket betyder att om en URL är http eller https kommer denna avlägsnas.

“Mentions” avlägsnas med RU `'@[A-Za-z0-9]+'`, d.v.s om en alfanumerisk sträng börjar med ‘@’ avlägsnas den alfanumeriska strängen.

Avlägsning av “hashtags”, skiljetecken, siffror och emojis görs med RU `'[^a-zA-Z]'`, d.v.s en sträng måste börja på bokstäver mellan a-z, om inte tas dessa bort.

Och slutligen avlägsnas repeterande mellanslag, vilket utförs med RU `' +'`. Ifall det är två eller fler repeterande mellanslag tas dessa bort.

Kodavsnitt 3.5: Funktioner för att avlägsna URL:er, ”mention”, ”hashtags”, skiljetecken, siffror, emojis och repeterade mellanslag.

```
def remove_links(self):
       self.df.loc[:, "tweet"] = self.df.loc[:, "tweet"].\
       replace(r'https?://[A-Za-z0-9./]+', '', regex=True)

def remove_twitter_mention(self):
       self.df.loc[:, "tweet"] = self.df.loc[:, "tweet"].\
       replace(r'@[A-Za-z0-9]+', '', regex=True)

def remove_hashtag(self):
       self.df.loc[:, "tweet"] = self.df.loc[:, "tweet"].\
       replace('[^a-zA-Z]', ' ', regex=True)

def remove_extra_whitepsace(self):
      self.df.loc[:, "tweet"] = self.df.loc[:, "tweet"].\
       replace(' +', ' ', regex=True)
```

För att utföra ordstamsigenkänning och avlägsning av stoppord måste varje twitterinlägg först tokeniseras, vilket betyder att varje ord i ett twitterinlägg blir ett eget element i en lista. Tokeniseringen utförs m.h.a funktionen `word_tokenize()` [61] från Natural Language Toolkit (NLTK), se kodavsnitt 3.6.

Kodavsnitt 3.6: Funktion för att tokenisera twitterinlägg.

```
def tokenize(self):
      self.df["tweet"] = self.df["tweet"].apply(word_tokenize)
```

I kodavsnitt 3.7 utförs ordstamsigenkänning m.h.a `SnowballStemmer('english').stem()` [62] från NLTK som försöker hitta varje ords ordstam vilket diskuteras i avsnitt 3.1.

Kodavsnitt 3.7: Funktion för att utföra ordstamsigenkänning.

```
def word_stemming(self):
       self.df["tweet"] = self.df["tweet"]./
       apply(lambda x: [SnowballStemmer('english').stem(y) for y in x])
```

Avlägsning av stoppord utförs m.h.a `stopwords.word('english')` [63] från NLTK som är en lista av engelska stoppord. `Remove_stopwords()` kontrollerar om något ord i twitterinlägget finns med i listan, om det gör det tas det bort, se kodavsnitt 3.8.

Kodavsnitt 3.8: Funktion för att avlägsna stoppord.

```
stop = stopwords.words('english')
def remove_stopwords(self):
        self.df["tweet"] = self.df["tweet"]./
        apply(lambda x: [word for word in x if word not in stop])
```

3.4.2 Lexikon

I detta avsnitt diskuteras hur den lexikonbaserade modellen implementeras som blir ett basfall för STS- och SemEval-datamängden. Tekniker som använts för implementationen är WordNet [64] och SentiWordNet [65] från python-paketet NLTK. WordNet är en lexikal databas på engelska som hittar konceptuella relationer mellan ord där substantiv, verb, adjektiv och adverb grupperas till mängder av kognitiva synonymer. SentiWordNet använder sedan de kognitiva synonymerna för att klassificera om de är positiva eller negativa.

I kodavsnitt 3.9 illustreras implementationen av den lexikonbaserade modellen, som tar ett twitterinlägg som parameter. Twitterinlägget delas sedan ner till en mängd ord med respektive POS-tag. Därefter utförs tre kontroller på respektive ord. Den första kontrollen kontrollerar om ordet är ett substantiv, adjektiv eller ett adverb. Sedan kontrolleras om ordet har en grundform och slutligen kontrolleras om ordet är en synonym. Om ordet inte klarar de tre kontrollerna hoppar vi över ordet, eftersom ordet inte kan klassificeras. Ifall ordet går igenom kontrollerna tas ordets klassificering fram och adderas till en variabel *sentiment*. När alla mängdens ord antingen har klassificerats eller hoppats över används *sentiment* för att klassificera om twitterinlägget är positivt eller negativt, d.v.s om $sentiment >= 0$ är inlägget positivt annars negativt.

Kodavsnitt 3.9: Funktion för att klassificera twitterinlägg m.h.a WordNet och SentiWordNet.

```
def swn_classifier(tweet):
    sentiment = 0.0
    tokens_count = 0

    sentinized_tweet = sent_tokenize(tweet)

    for sentence in sentinized_tweet:
        # Part of Speach tag
        tagged_tweet = pos_tag(word_tokenize(sentence))

        for word, tag in tagged_tweet:
            wn_tag = convert_tag(tag)
            if wn_tag not in (wn.NOUN, wn.ADJ, wn.ADV):
                continue

            lemma = lemmatizer.lemmatize(word, pos=wn_tag)
            if not lemma:
                continue

            synset = wn.synsets(lemma, pos=wn_tag)
            if not synset:
                continue

            synset = synset[0]
            swn_synset = swn.senti_synset(synset.name())

            sentiment += swn_synset.pos_score() - swn_synset.neg_score()
            tokens_count += 1

    if not tokens_count:
        return 0

    if sentiment >= 0:
        return 1

    return 0
```

Kodavsnitt 3.9 körs på både STS- och SemEval-datamängden och resultatet presenteras i avsnitt 4.

3.4.3 Naive Bayes

I detta avsnitt diskuteras hur Naive bayes-algoritmen implementerats och hur vi anpassat algoritmens parametrar för att uppnå en eventuellt förbättrad precision. För implementation av NB har SciKit-learn (sk-learn) [66] använts, som är ett Python-paket för maskininlärning.

För att implementera och träna Naive bayes-algoritmen måste först datamängden delas in i en träningsmängd och en testmängd. Delningen utförs för att möjliggöra utvärdering av algoritmens precision, vilket testmängden används för. Delningen utförs m.h.a sk-learn funktionen `train_test_split()` [67] där datamängden delas till 20% testmängd och 80% träningsmängd, se kodavsnitt 3.10.

Kodavsnitt 3.10: Kod för att dela upp datamängden i en träningsmängd och en testmängd.

```
X_train, X_test, y_train, y_test = train_test_split(df.tweet,
            df.lable, test_size=0.2, random_state=0)
```

I kodavsnitt 3.11 implementeras algoritmen `MultinomialNB()` med standardparametrar [68] och unigram som särdrag vilket är standardparameter för `CountVectorizer()` [69]. Träningen av algoritmen utförs med `fit(X_train, y_train)`-funktionen som tar träningsmängden som indata. Precisionen tas fram genom att först klassificera vår testmängd på den tränade algoritmen m.h.a `predict(X_test)` för att sedan använda `metrics.classification_report()` som presenterar värden som diskuterats i avsnitt 2.5.

Kodavsnitt 3.11: Kod för att skapa och träna NB-klassificeraren.

```
nb_unigram_clf = Pipeline([('vect', CountVectorizer()), ('clf', MultinomialNB())])
nb_unigram_clf.fit(X_train, y_train)
predicted = nb_unigram_clf.predict(X_test)
print(metrics.classification_report(y_test, predicted, target_names=target_names))
```

Kodavsnitt 3.11 körs på både STS- och SemEval-datamängden och resultatet presenteras i avsnitt 4.

I kodavsnitt 3.12 används `GridSearchCV()` [70] vilket är en sk-learn funktion för att utföra parametersökning för algoritmer och särdragsfunktioner så att optimala parametrar kan hittas. För `CountVectorizer()` är vi intresserade av `max_df`

och `ngram_range`. `Max_df` tar emot ett flyttal mellan intervallet $[0.0, 1.0]$ som under uppbyggnaden av vokabuläret ignorerar ord som har en twitterinläggsfrekvens högre än det angivna flyttalet. `Ngram_range` används för att testa vilken typ av ngram som skall användas, t.e.x (1, 1) betyder unigram och (1, 2) betyder att både uni- och bigrams används.

För `TfidfTransformer()` är vi intresserade av `use_idf` och `Norm`. `Use_idf` används för att specificera om IDF skall användas eller ej. `Norm` specificerar vilken typ av normaliseringsfunktion som används för att normalisera vårt ord. `SelectKBest()` [71] tar två parametrar, `score_func` och k. För `score_func` är vi bara intresserade av "chi-squared test" (chi2) och k är ett heltal för att specificera hur många särdrag som ska tas ut från `score_func`. Eftersom MNB är en så pass enkel klassificeringsalgoritm finns det inga parametrar att justera.

Kodavsnitt 3.12: Kod som används till parametersökning för NB.

```
pipeline = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('kbest', SelectKBest(chi2)),
    ('clf', MultinomialNB()),
])
parameters = {
    'vect__max_df': (0.5, 0.75, 1.0),
    'vect__ngram_range': ((1, 1), (1, 2), (1, 3)),  # uni-, bi-, trigram
    'tfidf__use_idf': (True, False),
    'tfidf__norm': (None, 'l1', 'l2'),
    'kbest__k': (x),
}

grid_search = GridSearchCV(pipeline, parameters, n_jobs=-1, verbose=1)
```

Kodavsnitt 3.12 kör både STS- och SemEval-datamängderna, där `'kbest__k': (x)` är den parameter som differentierar mellan datamängderna. I tabell 3.5 finns antal särdrag som vardera datamängd har och vilket intervall (x) sätts till för vardera datamängd. Resultatet av körningarna presenteras i avsnitt 4.

Tabell 3.5: Antal särdrag för STS- och SemEval-datamängderna och vilket intervall vi använder för parametersökning.

	Antal särdrag	Intervall
SemEval	10272	(3000, 6000, 8000, 'all')
STS	200000	(30000, 50000, 100000, 130000, 'all')

3.4.4 Support Vector Machine

I detta avsnitt diskuteras hur SVM-algoritmen implementerats och hur vi anpassat algoritmens parametrar för att uppnå en eventuellt förbättrad precision. SVM har också implementerats m.h.a sk-learn.

I avsnitt 3.4.3 diskuteras att en datamängd först måste delas in i en test- och träningsmängd, vilket även är fallet för SVM. Uppdelningen sker på samma sätt som illustreras i kodavsnitt 3.10.

I kodavsnitt 3.13 används algoritmen `SGDClassifier()` med standardparametrar [72]. `SGDClassifier()` är en linjär klassificerare baserat på en SVM med "stochastic gradient descent" (SGD) träning. Valet att använda `SGDClassifier()` istället för en ren linjär SVM är p.g.a att `SGDClassifier()` är bättre anpassad för stora datamängder [73], vilket är fallet för studien då STS-datamängden används. Som särdrag används unigram vilket är standardparametern för `CountVectorizer()`.

Träningen av algoritmen utförs med funktionen `fit(X_train, y_train)`, som tar träningsmängden som indata. Precisionen tas fram genom att klassificera testmängden på den tränade algoritmen m.h.a `predict(X_test)`. Sedan används `metrics.classification_report()` för att presentera utvärderingsvärdena som diskuteras i avsnitt 2.5.

Kodavsnitt 3.13: Kod för att skapa och träna SVM-klassificeraren.

```
svm_unigram_clf = Pipeline([
  ('vect', CountVectorizer()),
  ('clf', SGDClassifier())
])
svm_unigram_clf.fit(X_train, y_train)
predicted = svm_unigram_clf.predict(X_test)
```

```
print(metrics.classification_report(y_test, predicted, target_names=target_names))
```

För parametersökningen som illustreras i kodavsnitt 3.14 används `GridSearchCV()`. Vi är intresserade av samma parametrar för `CountVectorizer()`, `TfidfTransformer()` och `SelectKBest()` som diskuteras i avsnitt 3.4.3. För klassificeringsalgoritmen `SGDClassifier()` är följande parametrar av intresse: `max_iter`, `penalty` och `alpha`. `Max_iter` är hur många gånger algoritmen går igenom datamängden, `penalty` specificerar vilken typ av normaliseringsfunktion som används för att undvika överanpassning [74]. Överanpassning är ett fenomen där klassificeringsalgoritmen under träning lär sig detaljerna i datamängden vilket kommer skada precisionen när ny data klassificeras. `Alpha` används för att specificera vilken initial inlärningsfrekvens algoritmen använder. Resultatet av parametersökningen presenteras i avsnitt 4.

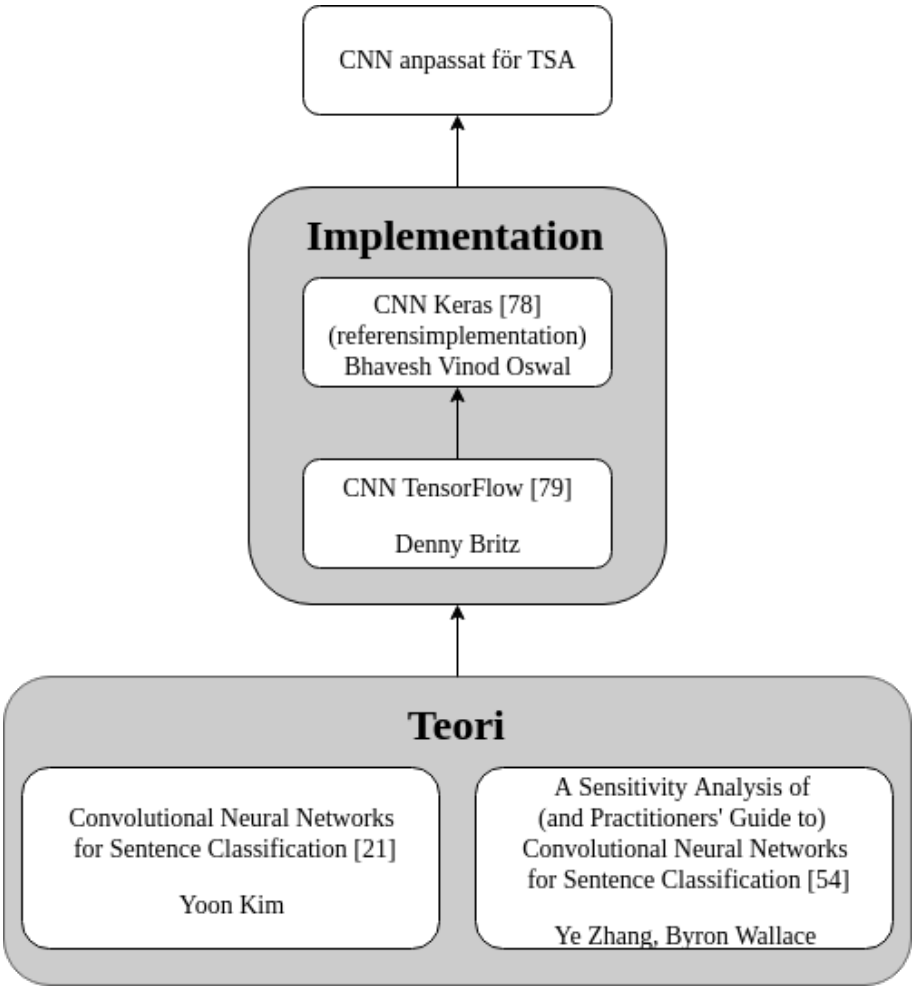
Kodavsnitt 3.14: Kod som används till parametersökning för SVM.

```
pipeline = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('kbest', SelectKBest(chi2)),
    ('clf', SGDClassifier()),
])
parameters = {
    'vect__max_df': (0.5, 0.75, 1.0),
    'vect__ngram_range': ((1, 1), (1, 2), (1, 3)),
    'tfidf__use_idf': (True, False),
    'tfidf__norm': (None, 'l1', 'l2'),
    'kbest__k': (x),
    'clf__alpha': (1e-4, 1e-3, 1e-2, 1e-1, 1e0, 1e1),
    'clf__max_iter': (500, 1000, 1500),
    'clf__penalty': ('l2', 'l1', 'elasticnet'),
}

grid_search = GridSearchCV(pipeline, parameters, n_jobs=-1, verbose=1)
```

3.4.5 Convolutional neural network

För implementationen av CNN har en referensimplementation använts, som har anpassats för att analysera twitterinlägg. Referensimplementationen har implementerats med Keras [75]. Keras är ett högnivåbibliotek för djupinlärning skrivet i Python, som utvecklades för att göra det snabbt och enkelt att experimentera med neuronnät. Det kan bland annat köras ovanpå TensorFlow [76], som är ett "open-source"-bibliotek för numerisk beräkning med starkt stöd för maskininlärning och djupinlärning.



Figur 3.7: Implementationer och teori som studiens CNN har baserats på.

I figur 3.7 visas hur implementationen av studiens CNN har tagits fram. I [77]

presenteras referensimplementationen, som är implementerad av Oswal. Referensimplementationen är en förenklad implementation baserad på Britz implementation [78]. Britz implementation är gjord med TensorFlow, i [79] presenteras och förklaras implementationen. Implementationen är baserad på Kims modell [21], som nämndes i avsnitt 2.2.3, och det Zhang och Wallace presenterar i [53]. Britz har förenklat modellen något, exempelvis används inte tränande ordinbäddningar utan istället används "one-hot"-vektorer för att representera ord.

Parametrarna som används i referensimplementationen skiljer sig dock från standardparametrarna som används i Britz implementation. För implementationen i den här studien har standardparametrarna från Britz implementation använts istället för de som används i referensimplementationen av Oswal. I kodavsnitt 3.15 presenteras parametrar som har använts för denna studie. Parametrarna används bland annat för att bestämma storleken på twitterinläggsmatrisen, storleken på de filter som ska användas och antalet filter för varje regionstorlek. Dessa beskrevs tidigare i avsnitt 3.3.3. Höjden och bredden på twitterinläggsmatrisen representeras i kodavsnittet av `sequence_length =x.shape[1]` och `embedding_dim =128`.

Parametern `drop = 0.5` används i "Dropout"-lagret för att bestämma hur stor del av neuronnätets enheter, som slumpmässigt ska kopplas bort vid varje uppdatering under träningen [80]. "Dropout"-lagret används för att undvika överanpassning. Vid träningen av en modell anges hur många gånger modellen ska tränas på en given datamängd [81]. Detta kallas för epoker och i koden har det angetts genom parametern `ephocs = 100`. Parametern `batch_size = 64` anger hur många exempel från träningsdatamängden som neuronätet ska tränas på åt gången [81].

Kodavsnitt 3.15: Parameterurval för CNN.

```
sequence_length = x.shape[1]
vocabulary_size = len(vocabulary_inv)
embedding_dim = 128
filter_sizes = [3, 4, 5]
num_filters = 128
drop = 0.5
epochs = 100
batch_size = 64
```

Databearbetningen som används i referensimplementationen har ändrats till den som beskrivs i avsnitt 3.1. Detta för att modellerna ska kunna utvärderas efter

samma förutsättningar. Uppdelningen av datamängderna har utförts på samma sätt som diskuteras i 3.4.3

Koden i kodavsnitt 3.16 används för att skapa lagret som hanterar data som skickas in och twitterinläggsmatrisen som ska användas för att analysera twitterinläggen.

Kodavsnitt 3.16: Kod för att skapa indata-lagret och twitterinläggsmatrisen.

```
inputs = Input(shape=(sequence_length,), dtype='int32')

embedding = Embedding(input_dim=vocabulary_size, output_dim=embedding_dim,
      input_length=sequence_length)(inputs)

reshape = Reshape((sequence_length, embedding_dim, 1))(embedding)
```

I kodavsnitt 3.17 skapas tre konvolutionslager. Arkitekturen är densamma som illustreras i 3.6, men istället för att det används enbart två filter för varje filterstorlek används istället `num_filters = 128` filter för respektive filterstorlek i implementation.

Kodavsnitt 3.17: Kod för att skapa konvulsionslager.

```
conv_0 = Conv2D(num_filters, kernel_size=(filter_sizes[0], embedding_dim),
    padding='valid', kernel_initializer='normal', activation='relu')(reshape)

conv_1 = Conv2D(num_filters, kernel_size=(filter_sizes[1], embedding_dim),
    padding='valid', kernel_initializer='normal', activation='relu')(reshape)

conv_2 = Conv2D(num_filters, kernel_size=(filter_sizes[2], embedding_dim),
    padding='valid', kernel_initializer='normal', activation='relu')(reshape)
```

Sedan appliceras “pooling”-lager på respektive konvolutionslager, se kodavsnitt 3.18.

Kodavsnitt 3.18: Kod för att skapa och applicera ”pooling”-lager på respektive konvolutionslager.

```
maxpool_0 = MaxPool2D(pool_size=(sequence_length - filter_sizes[0] + 1, 1),
    strides=(1,1), padding='valid')(conv_0)

maxpool_1 = MaxPool2D(pool_size=(sequence_length - filter_sizes[1] + 1, 1),
    strides=(1,1), padding='valid')(conv_1)
```

```
maxpool_2 = MaxPool2D(pool_size=(sequence_length - filter_sizes[2] + 1, 1),
    strides=(1,1), padding='valid')(conv_2)
```

Det som skickas ut från "pooling"-lagrena sätts samman till en ny särdragsvektor genom `Concatenate(axis=1)([maxpool_0, maxpool_1, maxpool_2])`. Ett fullt anslutet lager skapas med `Dense()` [80], som tar in den sammansatta särdragsvektorn och utför klassificeringen av twitterinlägg. Innan klassificeringen, appliceras först `Flatten()` [80] och sedan `Dropout()` [80] på särdragsvektorn, se kodavsnitt 3.19. `Flatten()` används för att ändra representationen av särdragsvektorn och `Dropout()` används för att undvika överanpassning.

Kodavsnitt 3.19: Kod för att sätta samman en ny särdragsvektor och koppla in ett fullt anslutet lager.

```
concatenated_tensor = Concatenate(axis=1)([maxpool_0, maxpool_1, maxpool_2])
flatten = Flatten()(concatenated_tensor)
dropout = Dropout(drop)(flatten)
output = Dense(units=2, activation='softmax')(dropout)
```

Modellen skapas med klassen `Model()` [82]. För att konfigurera modellen inför träning används metoden `compile()` [82] och för träna modellen används metoden `fit()` [82]. Skapandet och träningen av modelllen i referensimplementationen illustreras i kodavsnitt 3.20

Kodavsnitt 3.20: Kod för att skapa och träna CNN-klassificeraren.

```
model = Model(inputs=inputs, outputs=output)

checkpoint = ModelCheckpoint('weights.{epoch:03d}-{val_acc:.4f}.hdf5',
    monitor='val_acc', verbose=1, save_best_only=True, mode='auto')

adam = Adam(lr=1e-4, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.0)

model.compile(optimizer=adam, loss='binary_crossentropy', metrics=['accuracy'])

model.fit(X_train, y_train, batch_size=batch_size, epochs=epochs, verbose=1,
    callbacks=[checkpoint], validation_data=(X_test, y_test))
```

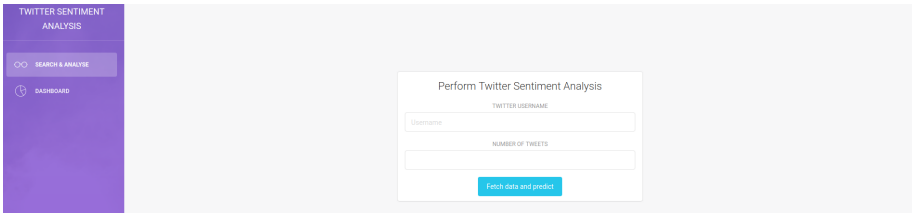
Zhang och Wallace förklarar i [53] att en negativ sida av CNN är att utvecklare måste ha kunskap nog att specificera den exakta arkitekturen av modellen och

dessutom ställa in alla medföljande parametrar. Parametrarna är så pass många till antalet att det kan vara överväldigande för en utvecklare som nyligen introducerats för tekniken. Att undersöka olika typer av konfigurationer av dessa parametrar är dessutom väldigt kostsamt. Dels för att det tar lång tid att träna den här typen av modeller, men även pga att antalet möjliga inställningar av dessa parametrar är så pass många. Av de anledningarna har ingen parametersökning utförts på CNN. Därför har enbart en CNN-modell tagits fram för detta experiment, med standardparametrarna från Britz implementation [78].

3.4.6 Webbgränssnitt

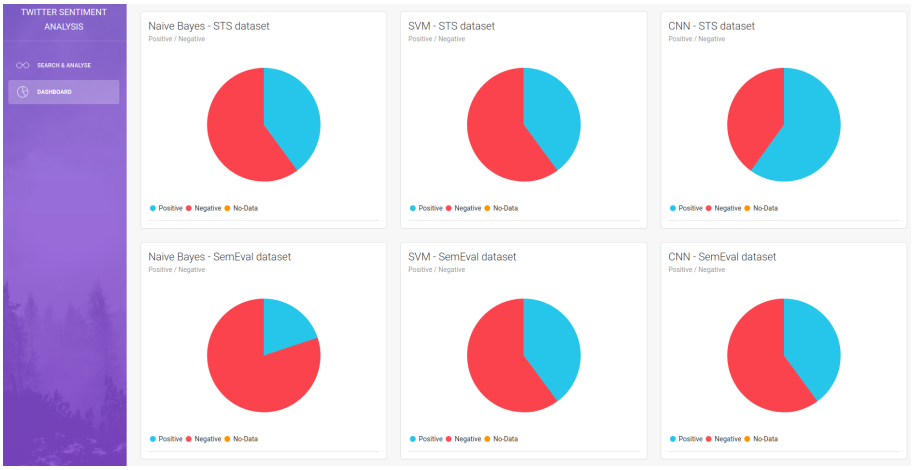
Som ett alternativ till studiens syfte hade uppdragsgivaren ett önskemål av en webbsida där en dynamisk sökning kunde utföras på en Twitter-användare, för att sedan presentera hur klassiciferingsmodellerna klassificerar Twitter-användarens twitterinlägg. Tekniker som valts för webbsidans gränssnitt är Bootstrap [83] och för webbsidans bakände används Flask [84] och SQLite3 [85]. Bootstrap är ett "open source" verktyg framtaget av Twitter för att bygga responsiva, mobila projekt på webben med HyperText Markup Language (HTML), Cascading Style Sheets (CSS) och Javascript (JS). Flask är ett mikro webbramverk som används för att prata med bakändefunktioner och rendera webbsidans gränssnitt. För att lagra resultaten från sökningen har SQLite3 används som databashanterare.

I figur 3.8 illustreras implementationen av webbgränssnittet där användare kan söka på en Twitter-användare och hur många twitterinlägg som ska klassificeras.



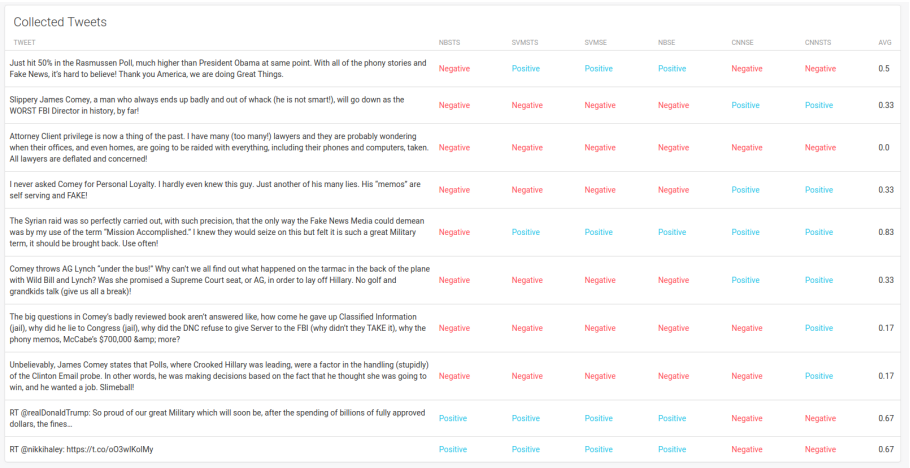
Figur 3.8: Illustrerar webbgränssnittets sökformulär.

I figur 3.9 illustreras en pajgraf för vardera klassificeringsmodell som visar hur många av de inhämtade twitterinläggen från Twitter-användaren "realDonald-Trump" som är positiva eller negativa.



Figur 3.9: Illustrerar hur klassificeringsmodellerna har klassificerat 10 stycken twitterinlägg från användaren “realDonaldTrump” som är inhämtade 2018-04-16 11:40.

Figur 3.10 illustrerar de inhämtade twitterinläggen från användaren “realDonaldTrump” i en tabell där användare kan analysera hur varje klassificeringsmodell har klassificerat varje twitterinlägg.



Collected Tweets

TWEET	NBSTS	SVMSTS	SVMSE	NBSE	CNNSE	CNNSTS	AVG
Just hit 50% in the Rasmussen Poll, much higher than President Obama at same point. With all of the phony stories and Fake News, it's hard to believe! Thank you America, we are doing Great Things.	Negative	Positive	Positive	Positive	Negative	Negative	0.5
Slippery James Comey, a man who always ends up badly and out of whack (he is not smart!), will go down as the WORST FBI Director in history, by far!	Negative	Negative	Negative	Negative	Positive	Positive	0.33
Attorney Client privilege is now a thing of the past. I have many (too many!) lawyers and they are probably wondering when their offices, and even homes, are going to be raided with everything, including their phones and computers, taken. All lawyers are deflated and concerned!	Negative	Negative	Negative	Negative	Negative	Negative	0.0
I never asked Comey for Personal Loyalty. I hardly even knew this guy. Just another of his many lies. His "memos" are self serving and FAKE!	Negative	Negative	Negative	Negative	Positive	Positive	0.33
The Syrian raid was so perfectly carried out, with such precision, that the only way the Fake News Media could demean was by my use of the term "Mission Accomplished." I knew they would seize on this but felt it is such a great Military term, it should be brought back. Use often!	Negative	Positive	Positive	Positive	Positive	Positive	0.83
Comey throws AG Lynch "under the bus!" Why can't we all find out what happened on the tarmac in the back of the plane with Wild Bill and Lynch? Was she promised a Supreme Court seat, or AG, in order to lay off Hillary. No golf and grandkids talk (give us all a break)!	Negative	Negative	Negative	Negative	Positive	Positive	0.33
The big questions in Comey's badly reviewed book aren't answered like, how come he gave up Classified Information (jail), why did he lie to Congress (jail), why did the DNC refuse to give Server to the FBI (why didn't they TAKE it), why the phony memos, McCabe's $700,000 & more?	Negative	Negative	Negative	Negative	Negative	Positive	0.17
Unbelievably, James Comey states that Polls, where Crooked Hillary was leading, were a factor in the handling (stupidly) of the Clinton Email probe. In other words, he was making decisions based on the fact that he thought she was going to win, and he wanted a job. Slimeball!	Negative	Negative	Negative	Negative	Negative	Positive	0.17
RT @realDonaldTrump: So proud of our great Military which will soon be, after the spending of billions of fully approved dollars, the fines...	Positive	Positive	Positive	Positive	Negative	Negative	0.67
RT @nikkihaley: https://t.co/oO3wlKolMy	Positive	Positive	Positive	Positive	Negative	Negative	0.67

Figur 3.10: Illustrerar hur klassificeringsmodellerna har klassificerat varje tweet.

3.5 Sammanfattning

I detta kapitel har tillvägagångssättet för studiens experiment diskuterats. En teknisk fördjupning har presenterats för databearbetning, särdragsurval och de tre algoritmerna, NB, SVM och CNN. Utöver detta presenteras även implementeringen av databearbetning, särdragsurval och de tre algoritmerna, där även ett webbgränssnitt har tagits fram efter önskemål från uppgradsgivaren.

Kapitel 4

Resultat

I detta kapitel kommer resultaten från studien att presenteras. Det kommer att presenteras hur lexikon, NB, SVM och CNN presterar när de har tränats med datamängderna som diskuterades i avsnitt 2.3. För lexikon och CNN har en modell tagits fram för respektive datamängd, medan två modeller har tagits fram för NB och SVM. Modellerna som har tagits fram för NB och SVM har tränats med två olika uppsättningar av parametrar. Den ena modellen har tränats med standardparametrar, medan den andra har tränats med parametrar framtagna m.h.a parametersökning. Av den anledningen kommer resultaten för NB och SVM omfatta: prestandan för de två modeller och parametrarna som togs fram genom parametersökningen. Till skillnad från lexikon och CNN där resultaten enbart omfattar en modell. Den lexikonbaserade modellen används som ett basfall för respektive datamängd, vilket diskuterades i kapitel 3.

För- och nackdelar med respektive klassificeringsmodell och datamängd kommer att diskuteras med hänsyn till prestanda, hur invecklade de var att implementera och hur lång tid de tog att träna. Värdena som kommer att användas för att jämföra och analysera prestandan av de olika modellerna diskuterades i avsnitt 2.5. I avsnitt 4.1 och 4.2 presenteras resultaten av prestandan för klassificeringsmodellerna, för respektive datamängd. Och i avsnitt 4.3 utvärderas klassificeringsmodellerna och datamängderna utefter för- och nackdelar.

4.1 Stanford Twitter Sentiment

Detta avsnitt avser att presentera och jämföra klassificeringsmodellerna som är tränade på STS-datamängden.

4.1.1 Lexikon

I tabell 4.1 presenteras prestandan efter att den lexikonbaserade modellen har tränats på STS-datamängden. Den lexikonbaserade modellen uppnår en precision på 59.6%, återkallelse på 58.5% och F-Score på 57.3%. Med detta i åtanke kan vi konstatera att basfallet precision är 9.6% bättre än att gissa om ett twitterinlägg är positivt eller negativt.

Tabell 4.1: Prestanda för lexikonbaserade modellen tränad på STS-datamängden.

	Precision	**Återkallelse**	**F-Score**	**Antal twitterinlägg**
Positiv	0.628	0.415	0.500	159815
Negativ	0.564	0.754	0.645	160185
Totalt genomsnitt	0.596	0.585	0.573	320000

4.1.2 Naive Bayes

I tabell 4.2 presenteras prestandan efter att NB med standardparamterar har tränats på STS-datamängden. Resultatet visar att NB med standardparametrar uppnår 77.1% i precision, återkallelse och F-Score vilket är en höjning på respektive 17.5%, 18.6% och 19.8% från datamängdens basfall.

Tabell 4.2: Prestanda för NB med standardparametrar tränad på STS-datamängden.

	Precision	**Återkallelse**	**F-Score**	**Antal twitterinlägg**
Positiv	0.761	0.789	0.775	159815
Negativ	0.781	0.753	0.767	160185
Totalt genomsnitt	0.771	0.771	0.771	320000

I tabell 4.3 presenteras de parametrar som blivit framtagna m.h.a parametersökningen som kodavsnitt 3.12 kör på STS-datamängden. Med de framtagna parametrarna har den andra NB modellen tagits fram.

Tabell 4.3: Resultat efter parametersökning för NB på STS-datamängden.

Optimala parametrar	
vect__max_df	0.5
vect__ngram_range	(1, 3)
tfidf__use_idf	False
tfidf__norm	'l2'
kbest__k	'all'

Tabell 4.4 presenterar prestandan efter att den andra NB-modellen har tränats på STS-datamängden. Resultatet visar att den tränade klassificeringsmodellen uppnår en precision på 79.2%, återkallelse på 79.0% och en F-Score på 78.9%. Detta betyder att vi uppnår en ökning på respektive 2.1%, 1.9% och 1.8% gentemot NB med standardparametrar.

Tabell 4.4: Prestanda för NB med optimala parametrar tränad på STS-datamängden.

	Precision	**Återkallelse**	**F-Score**	**Antal twitterinlägg**
Positiv	0.767	0.833	0.798	159435
Negativ	0.781	0.753	0.767	159334
Totalt genomsnitt	0.792	0.790	0.789	318769

4.1.3 Support Vector Machine

I tabell 4.5 presenteras prestandan efter att SVM med standardparamterar har tränats på STS-datamängden. Resultatet visar att den tränade klassificeringsmodellen uppnår en precision på 77.7%, en återkallelse och F-Score på 77.5% vilket är en höjning på respektive 18.1%, 19% och 20.2% jämfört med datamängdens basmodell.

Tabell 4.5: Prestanda för SVM med standardparametrar tränad på STS-datamängden.

	Precision	**Återkallelse**	**F-Score**	**Antal twitterinlägg**
Positiv	0.801	0.732	0.765	159815
Negativ	0.754	0.818	0.785	160185
Totalt genomsnitt	0.777	0.775	0.775	320000

I tabell 4.3 presenteras de parametrar som blivit framtagna m.h.a parametersökningen som kodavsnitt 3.14 kör på STS-datamängden. Med de framtagna parametrarna har den andra SVM modellen tagits fram.

Tabell 4.6: Resultat av parametersökning för NB på STS.

Optimala parametrar	
vect__max_df	0.5
vect__ngram_range	(1, 3)
tfidf__use_idf	True
tfidf__norm	None
kbest__k	'all'
clf__alpha	0.001
clf__max_iter	1500
clf__penalty	'l2'

Tabell 4.7 presenterar prestandan efter att den andra SVM modellen har tränats på STS-datamängden. Resultatet visar att den tränade klassificeringsmodellen uppnår en precision på 78.6%, återkallelse och F-Score på 78.4%. Detta betyder att vi uppnår en ökning på 0,9% för respektive värde gentemot SVM med standardparametrar.

Tabell 4.7: Prestanda för SVM med optimala parametrar tränad på STS-datamängden.

	Precision	**Återkallelse**	**F-Score**	**Antal twitterinlägg**
Positiv	0.811	0.742	0.775	159815
Negativ	0.764	0.828	0.795	160185
Totalt genomsnitt	0.786	0.784	0.784	320000

4.1.4 Convolutional neural network

Prestandan för CNN-modellen, som har tränats på STS-datamängden, presenteras i tabell 4.8. Modellen uppnådde en precision, återkallelse och F-score på 80,4%. I jämförelse med datamängdens basfall är det en ökning på 20,8% i precision, 21,9% i återkallelse och 23,1% i F-score.

Tabell 4.8: Prestanda för CNN tränad på STS-datamängden.

	Precision	**Återkallelse**	**F-Score**	**Antal twitterinlägg**
Positiv	0.812	0.790	0.801	159435
Negativ	0.796	0.817	0.806	159334
Totalt genomsnitt	0.804	0.804	0.804	318769

4.2 SemEval

I detta avsnitt presenteras och jämförs de klassificeringsmodeller som har tränats på datamängden SemEval.

4.2.1 Lexikon

I tabell 4.9 visas prestandan för den lexikonbaserade modellen, som har tränats på SemEval-datamängden. Modellen uppnår en precision på 68,2%, en återkallelse på 67,1% och en F-Score på 67,6%.

Tabell 4.9: Prestanda för lexikonbaserade modellen tränad på SemEval-datamängden.

	Precision	**Återkallelse**	**F-Score**	**Antal twitterinlägg**
Positiv	0.457	0.510	0.482	618
Negativ	0.779	0.740	0.759	1440
Totalt genomsnitt	0.682	0.671	0.676	2058

4.2.2 Naive Bayes

Prestandan för NB-modellen med standardparametrar, som tränats på SemEval-datamängden, visas i tabell 4.9. NB-modellen uppnådde en precision på 85,4%, en återkallelse på 85,7% och en F-score på 85,2%. Det är en ökning på 17,2% i precision, 18,6% i återkallelse och 17,6% i F-score gentemot datamängdens basmodell.

Tabell 4.10: Prestanda för NB med standardparametrar tränad på STS-datamängden.

	Precision	**Återkallelse**	**F-Score**	**Antal twitterinlägg**
Positiv	0.820	0.670	0.737	618
Negativ	0.869	0.937	0.901	1440
Totalt genomsnitt	0.854	0.857	0.852	2058

I tabell 4.11 visas de parametrar som togs fram m.h.a parametersökningen. Parametersökningen utfördes med koden i kodavsnitt 3.12. Dessa parametrar användes för att ta fram den andra NB-modellen för SemEval-datamängden.

Tabell 4.11: Resultat av parametersökning för NB på SemEval.

Optimala parametrar	
vect__max_df	0.5
vect__ngram_range	(1, 1)
tfidf__use_idf	False
tfidf__norm	None
kbest__k	'8000'

Den andra NB-modellens prestanda presenteras i tabell 4.12. Modellen uppnådde en precision på 85,4%, återkallelse på 85,7% och F-Score på 85,2%. I jämförelse med modellen med standardparametrar resulterade parametersökningen i en lägre prestanda.

Tabell 4.12: Prestanda för NB med optimala parametrar tränad på SemEval-datamängden.

	Precision	**Återkallelse**	**F-Score**	**Antal twitterinlägg**
Positiv	0.802	0.676	0.734	618
Negativ	0.870	0.928	0.898	1440
Totalt genomsnitt	0.850	0.853	0.849	2058

4.2.3 Support Vector Machine

I tabell 4.13 prestenteras prestandan för SVM-modellen med standardparametrar som har tränats på SemEval-datamängden. Modellen uppnådde en precision på 84,9%, en återkallelse på 84,8% och en F-Score på 84,9%. En ökning på 16,7% i precision, 17,7% i återkallelse och F-score på 17,3% gentemot datamängdens basfall.

Tabell 4.13: Prestanda för SVM med standardparametrar tränad på SemEval-datamängden.

	Precision	**Återkallelse**	**F-Score**	**Antal twitterinlägg**
Positiv	0.742	0.759	0.750	618
Negativ	0.896	0.887	0.891	1440
Totalt genomsnitt	0.849	0.848	0.849	2058

Parametrarna som togs fram m.h.a parametersökning för den andra SVM-modellen presenteras i tabell 4.14. Parametersökningen utfördes med koden i kodavsnitt 3.14.

Tabell 4.14: Resultat av parametersökning för SVM på SemEval.

Optimala parametrar	
vect__max_df	0.75
vect__ngram_range	(1, 2)
tfidf__use_idf	True
tfidf__norm	'l2'
kbest__k	'all'
clf__alpha	0.0001
clf__max_iter	1000
clf__penalty	'l2'

Prestandan för den andra SVM-modellen presenteras i tabell 4.15. Modellen uppnådde en precision och återkallelse på 87,1% samt 86,5% i F-Score. Detta resulterar i en ökning på 2,2% i precision, 2,3% i återkallelse och 1,6% i F-Score gentemot SVM med standardparametrar.

Tabell 4.15: Prestanda för SVM med optimala parametrar tränad på SemEval-datamängden.

	Precision	**Återkallelse**	**F-Score**	**Antal twitterinlägg**
Positiv	0.878	0.662	0.755	618
Negativ	0.869	0.960	0.912	1440
Totalt genomsnitt	0.871	0.871	0.865	2058

4.2.4 Convolutional neural network

Prestandan för CNN-modellen, som har tränats på SemEval-datamängden, presenteras i tabell 4.16. Modellen uppnådde en precision på 86.5%, återkallelse på 86.7% och en F-Score på 86.5% vilket är en höjning på respektive 18.3%, 19.6% och 18.9% gentemot datamängdens basfall.

Tabell 4.16: Prestanda för CNN tränad på SemEval-datamängden.

	Precision	**Återkallelse**	**F-Score**	**Antal twitterinlägg**
Positiv	0.812	0.727	0.767	618
Negativ	0.888	0.928	0.907	1440
Totalt genomsnitt	0.865	0.867	0.865	2058

4.3 Utvärdering

4.3.1 Prestanda

4.3.2 Träning

4.3.3 Implementation

4.4 Sammanfattning

Kapitel 5

Slutsats

5.1 Sammanfattning

5.2 Problem

5.3 Begränsningar

5.4 Vidare utveckling

5.5 Slutord

Litteraturförteckning

[1] Hoda Korashy Walaa Medhat, Ahmed Hassan. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, Volume 5(Issue 4):1093–1113, December 2014. URL `https://ac.els-cdn.com/S2090447914000550/1-s2.0-S2090447914000550-main.pdf?_tid=77d36d1a-ff80-11e7-956d-00000aacb362&acdnat=1516631469_c14e5bc49162e2b9a4232c2931592298`.

[2] Anastasia Giachanou and Fabio Crestani. Like it or not: A survey of twitter sentiment analysis methods. *ACM Comput. Surv.*, 49(2):28:1–28:41, June 2016. ISSN 0360-0300. doi: 10.1145/2938640. URL `http://doi.acm.org/10.1145/2938640`.

[3] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[4] Wikipedia contributors. Atari games — Wikipedia, the free encyclopedia, 2016. URL `https://en.wikipedia.org/wiki/Atari_Games`. [Online; accessed 5-Februari 2018].

[5] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[6] Wikipedia contributors. Lee sedol — Wikipedia, the free encyclopedia,

2016. URL https://en.wikipedia.org/wiki/Lee_Sedol. [Online; accessed 5-Februari 2018].

[7] Wikipedia contributors. Go(game) — Wikipedia, the free encyclopedia, 2016. URL https://en.wikipedia.org/wiki/Go_(game). [Online; accessed 5-Februari 2018].

[8] Apple. Ios - siri - apple, n.d. URL https://www.apple.com/ios/siri/. [Online; accessed 5-March 2018].

[9] Google. Google assistant - your own personal google, n.d. URL https://assistant.google.com/. [Online; accessed 5-March 2018].

[10] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708, 2014.

[11] Johan Ugander and Lars Backstrom. Balanced label propagation for partitioning massive graphs. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 507–516. ACM, 2013.

[12] Wikipedia contributors. Unsupervised learning — Wikipedia, the free encyclopedia, 2016. URL https://en.wikipedia.org/wiki/Unsupervised_learning. [Online; accessed 5-Februari 2018].

[13] Jason Brownlee. Supervised and unsupervised machine learning algorithms, 2016. URL https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/.

[14] Wikipedia contributors. Bayes' theorem — Wikipedia, the free encyclopedia, 2002. URL https://en.wikipedia.org/wiki/Bayes%27_theorem. [Online; accessed 12-February 2018].

[15] Sunil Ray. 6 easy steps to learn naive bayes algorithm (with codes in python and r), 2015. URL https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/. [Online; accessed 12-February 2018].

[16] Eric W. Weisstein. "hyperplane.from mathworld–a wolfram web resource, n.d. URL http://mathworld.wolfram.com/Hyperplane.html. [Online; accessed 12-February 2018].

[17] Noel Bambrick. Support vector machines: A simple explanation, n.d. URL https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html. [Online; accessed 12-February 2018].

[18] Wikipedia contributors. Support vector machine — Wikipedia, the free encyclopedia, 2002. URL https://en.wikipedia.org/wiki/Support_vector_machine. [Online; accessed 12-February 2018].

[19] S. Albawi, T. A. Mohammed, and S. Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, Aug 2017. doi: 10.1109/ICEngTechnol.2017.8308186.

[20] Wikipedia contributors. Artificial neural network — wikipedia, the free encyclopedia, 2018. URL https://en.wikipedia.org/w/index.php?title=Artificial_neural_network&oldid=830851868. [Online; accessed 28-March-2018].

[21] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[22] Shiyang Liao, Junbo Wang, Ruiyun Yu, Koichi Sato, and Zixue Cheng. Cnn for situations understanding based on sentiment analysis of twitter data. *Procedia Computer Science*, 111:376 – 381, 2017. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2017.06.037. URL http://www.sciencedirect.com/science/article/pii/S1877050917312103. The 8th International Conference on Advances in Information Technology.

[23] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12), 2009.

[24] International workshop on semantic evaluation 2017, 2016. URL http://alt.qcri.org/semeval2017/. [Online; accessed 12-February 2018].

[25] Sentiment analysis in twitter, 2016. URL http://alt.qcri.org/semeval2017/task4/. [Online; accessed 12-February 2018].

[26] Amazon. Amazon mechanical turk, n.d. URL https://www.mturk.com/product-details. [Online; accessed 12-February 2018].

[27] Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. Semeval-2016 task 4: Sentiment analysis in twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1–18, 2016.

[28] Wikipedia contributors. Emoji — Wikipedia, the free encyclopedia, 2014. URL https://sv.wikipedia.org/wiki/Emoji. [Online; accessed 5-Februari 2018].

[29] Wikipedia contributors. Twitter — Wikipedia, the free encyclopedia, 2007. URL https://en.wikipedia.org/wiki/Twitter. [Online; accessed 5-Februari 2018].

[30] Wikipedia contributors. Confusion matrix — Wikipedia, the free encyclopedia, 2014. URL https://en.wikipedia.org/wiki/Confusion_matrix. [Online; accessed 29-January 2018].

[31] Wikipedia contributors. Precision and recall — Wikipedia, the free encyclopedia, 2016. URL https://en.wikipedia.org/wiki/Precision_and_recall. [Online; accessed 29-January 2018].

[32] Wikipedia contributors. F1 score — Wikipedia, the free encyclopedia, 2014. URL https://en.wikipedia.org/wiki/F1_score. [Online; accessed 29-January 2018].

[33] Dimitrios Effrosynidis, Symeon Symeonidis, and Avi Arampatzis. A comparison of pre-processing techniques for twitter sentiment analysis. In *International Conference on Theory and Practice of Digital Libraries*, pages 394–406. Springer, 2017.

[34] Emma Haddi, Xiaohui Liu, and Yong Shi. The role of text pre-processing in sentiment analysis. *Procedia Computer Science*, 17:26 – 32, 2013. ISSN

1877-0509. doi: https://doi.org/10.1016/j.procs.2013.05.005. URL http://www.sciencedirect.com/science/article/pii/S1877050913001385. First International Conference on Information Technology and Quantitative Management.

[35] Z. Jianqiang and G. Xiaolin. Comparison research on text pre-processing methods on twitter sentiment analysis. *IEEE Access*, 5:2870–2879, 2017. ISSN 2169-3536. doi: 10.1109/ACCESS.2017.2672677.

[36] Wikipedia contributors. Feature (machine learning) — Wikipedia, the free encyclopedia, 2004. URL https://en.wikipedia.org/wiki/Feature_(machine_learning). [Online; accessed 12-Mars 2018].

[37] scikitlearn documentation. Working with text data, n.d. URL http://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html. [Online; accessed 12-Mars 2018].

[38] Cambridge University Press. Inverse document frequency, 2009-04-07. URL https://nlp.stanford.edu/IR-book/html/htmledition/inverse-document-frequency-1.html. [Online; accessed 11-April 2018].

[39] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12), 2009.

[40] Efthymios Kouloumpis, Theresa Wilson, and Johanna D Moore. Twitter sentiment analysis: The good the bad and the omg! *Icwsm*, 11(538-541):164, 2011.

[41] Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. Sentiment analysis of twitter data. In *Proceedings of the workshop on languages in social media*, pages 30–38. Association for Computational Linguistics, 2011.

[42] Rahul Saxena. How the naive bayes classifier works in machine learning, 2017. URL http://dataaspirant.com/2017/02/06/naive-bayes-classifier-machine-learning/. [Online; accessed 12-February 2018].

[43] Stephanie. Bernoulli distribution: Definition and examples, 2017. URL http://www.statisticshowto.com/bernoulli-distribution/. [Online; accessed 12-February 2018].

[44] scikitlearn documentation. 1.9 naive bayes – scikitlearn documentation, 2017. URL http://scikit-learn.org/stable/modules/naive_bayes.html. [Online; accessed 12-February 2018].

[45] Wikipedia contributors. Dirichlet distribution — Wikipedia, the free encyclopedia, 2004. URL https://en.wikipedia.org/wiki/Dirichlet_distribution. [Online; accessed 26-Februari 2018].

[46] David Heckerman. A tutorial on learning with bayesian networks. In *Learning in graphical models*, pages 301–354. Springer, 1998.

[47] Wikipedia contributors. Additive smoothing — Wikipedia, the free encyclopedia, 2008. URL https://en.wikipedia.org/wiki/Additive_smoothing#Pseudocount. [Online; accessed 26-Februari 2018].

[48] Bruno Stecanella. An introduction to support vector machines (svm), 2017. URL https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/. [Online; accessed 19-Mars 2018].

[49] Wikipedia contributors. Kernel method — Wikipedia, the free encyclopedia, 2005. URL https://en.wikipedia.org/wiki/Kernel_method. [Online; accessed 19-Mars 2018].

[50] Alexandre Kowalczyk. Linear kernel: Why is it recommended for text classification ?, 2014. URL https://www.svm-tutorial.com/2014/10/svm-linear-kernel-good-text-classification/. [Online; accessed 27-Mars 2018].

[51] Wikipedia contributors. Convolutional neural network — wikipedia, the free encyclopedia, 2018. URL https://en.wikipedia.org/w/index.php?title=Convolutional_neural_network&oldid=832961286. [Online; accessed 3-April-2018].

[52] Denny Britz. Understanding convolutional neural networks for nlp, 2015. URL http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/.

[53] Ye Zhang and Byron Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*, 2015.

[54] Adit Deshpande. A beginner's guide to understanding convolutional neural networks, 2016. URL `https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/`.

[55] Wikipedia contributors. Word embedding — Wikipedia, the free encyclopedia, 2018. URL `https://en.wikipedia.org/w/index.php?title=Word_embedding&oldid=835602312`. [Online; accessed 11-April-2018].

[56] Python 3.0 release, n.d. URL `https://www.python.org/download/releases/3.0/`. [Online; accessed 4-April 2018].

[57] pandas: powerful python data analysis toolkit¶, 2017. URL `https://pandas.pydata.org/pandas-docs/stable/`. [Online; accessed 4-April 2018].

[58] Leonard Richardson. Beautiful soup documentation, 2004. URL `https://www.crummy.com/software/BeautifulSoup/bs4/doc/#`. [Online; accessed 4-April 2018].

[59] 6.5. unicodedata — unicode database, n.d. URL `https://www.crummy.com/software/BeautifulSoup/bs4/doc/#`. [Online; accessed 4-April 2018].

[60] Jeffrey Friedl. 6.2. re — regular expression operations, 2009. URL `https://docs.python.org/3/library/re.html`. [Online; accessed 4-April 2018].

[61] nltk.tokenize package - nltk 3.2.5 documentation, n.d. URL `http://www.nltk.org/api/nltk.tokenize.html`. [Online; accessed 4-April 2018].

[62] nltk.stem package - nltk 3.2.5 documentation, n.d. URL `http://www.nltk.org/api/nltk.tokenize.html`. [Online; accessed 4-April 2018].

[63] Removing stop words with nltk in python - geeksforgeels, n.d. URL `http://www.nltk.org/api/nltk.tokenize.html`. [Online; accessed 4-April 2018].

[64] Princeton University. Princeton university äbout wordnet., 2010. URL `https://wordnet.princeton.edu/`. [Online; accessed 17-April 2018].

[65] SentiWordNet. Sentiwordnet, 2010. URL sentiwordnet.isti.cnr.it. [Online; accessed 17-April 2018].

[66] scikitlearn documentation. sci-kit learn - machine learning in python., n.d. URL http://scikit-learn.org/stable/index.html#. [Online; accessed 12-April 2018].

[67] scikitlearn documentation. sklearn.model-selection.train-test-split - sk-learn 0.19.1 documentation, n.d. URL http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html. [Online; accessed 11-April 2018].

[68] scikitlearn documentation. sklearn.naive-bayes.multinomialnb - sk-learn 0.19.1 documentation, n.d. URL http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html. [Online; accessed 11-April 2018].

[69] scikitlearn documentation. sklearn.feature-extraction.text.countvectorizer - sk-learn 0.19.1 documentation, n.d. URL http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html. [Online; accessed 11-April 2018].

[70] scikitlearn documentation. sklearn.model.selection.gridsearchcv - sk-learn 0.19.1 documentation, n.d. URL http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html. [Online; accessed 11-April 2018].

[71] scikitlearn documentation. sklearn.feature-selection.selectkbest - sk-learn 0.19.1 documentation, n.d. URL http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html. [Online; accessed 11-April 2018].

[72] scikitlearn documentation. sklearn.linear-model.sgdclassifier - sk-learn 0.19.1 documentation, n.d. URL http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html. [Online; accessed 11-April 2018].

[73] scikitlearn documentation. 1.4. support vector machines - sk-learn 0.19.1 documentation, n.d. URL http://scikit-learn.org/stable/modules/svm.html. [Online; accessed 11-April 2018].

[74] Jason Brownlee. Overfitting and underfitting with machine learning algorithms, 2016. URL https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/. [Online; accessed 16-April 2018].

[75] Keras documentation, n.d. URL https://keras.io/. [Online; accessed 16-April 2018].

[76] Tensorflow, n.d. URL https://www.tensorflow.org/. [Online; accessed 16-April 2018].

[77] Bhavesh Vinod Oswal. Cnn-text-classification-keras, 2016. URL https://github.com/bhaveshoswal/CNN-text-classification-keras. [Online; accessed 16-April 2018].

[78] Denny Britz. Convolutional neural network for text classification in tensorflow, 2015. URL https://github.com/dennybritz/cnn-text-classification-tf. [Online; accessed 16-April 2018].

[79] Denny Britz. Implementing a cnn for text classification in tensorflow, 2015. URL http://www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow/. [Online; accessed 16-April 2018].

[80] Core layers - keras documentation, n.d. URL https://keras.io/core/. [Online; accessed 16-April 2018].

[81] Sequential - keras documentation, n.d. URL https://keras.io/models/sequential/#sequential-model-methods. [Online; accessed 16-April 2018].

[82] Model (functional api) - keras documentation, n.d. URL https://keras.io/models/model/. [Online; accessed 17-April 2018].

[83] Bootstrap - the most popular html, css and js library in the world, n.d. URL https://getbootstrap.com/. [Online; accessed 16-April-2018].

[84] Armin Ronacher. Welcome - flask a python microframework, 2010. URL `http://flask.pocoo.org/`. [Online; accessed 16-April-2018].

[85] About sqlite, 2000. URL `http://flask.pocoo.org/`. [Online; accessed 16-April-2018].