Förbättrad direktöversättning

PROJEKT SPRÅKTEKNOLOGI

Filip Nääs Starberg & Axel Rooth

DD1418 HT20

Innehållsförteckning

1.Introduktion

Direktöversättning är ett högaktuellt område inom maskininlärning. Fram tills 2000-talet så användes statistiska språkmodeller till största delen, i många fall baserat på n-gram modeller. Dessa modeller användes bland annat av Google Translate[1]. Däremot var deras resultat blekt i jämförelse med de neurala nätverken som från 2010 och framåt är den metod man använder för att översätta språk. Fördelen med neurala nätverk inom språkteknologi är deras generalitet i den bemärkelsen att om modellen får tillräckligt med data finner modellen ett bra resultat. I jämförelse med statistiska modeller måste någon expert inom både översättningsspråk och målspråket justera den statistiska modellen tills resultatet blir godartat, vilket är både resurskrävande och tidskrävande.

2.Bakgrund

I detta projekt har två modeller skapats med två olika tillvägagångssätt för hur en direktöversättning kan åstadkommas. Bigram-modellen fungerar genom att man manuellt använder onlineöversättaren bab.la[2] och väljer ord på måfå för att översätta en svensk mening till engelska ord för ord utan hänsyn tagen till kontext. Därefter använder man alla alternativa ord bab.la ger för att sedan låta modellen försöka förbättra meningen. Modellen väljer den ordföljd som verkar mest sannolik givet det träningskorpus den tränats på.

Den andra modellen använder sig av en sekvens till sekvens-struktur (seq2seq) med hjälp av Long-Short Term Memory nätverk för att åstadkomma en direkt översättning mellan engelska till franska. Till skillnad från bigram-modellen sker inget mellansteg för översättningen utan modeller sköter detta arbete från start till slut. Dessutom är modellen betydligt mer användbar för översättning av längre sekvenser tack vare LSTM:s uppbyggnad.

3.Hypotes

3.1 N-Gram

Den inledande hypotesen var att en implementation med n-gram inte skulle ge bra resultat men att den skulle vara enkel att implementera och effektiv ur ett beräkningsperspektiv då den bygger på sannolikheter som är relativt simpla.

3.2 LSTM Encoder - Decoder modell

Med en brist på tidigare erfarenhet kring LSTM-nätverk var vår tidiga, och något naiva, hypotes att översättningen enbart skulle vara minimalt bristfällig. Därmed borde den kunna översätta engelska meningar till franska utan större bekymmer.

---

[1] "The remarkable way Google Translate actually works", in *The Independent*, , 2020, <https://www.independent.co.uk/life-style/gadgets-and-tech/news/google-translate-how-work-foreign-languages-interpreter-app-search-engine-a8406131.html> [accessed 13 December 2020].
[2] https://bab.la

4.Teori

Denna rubrik har lagts till för att förtydliga modellens struktur och för att få en djupare förståelse. På grund ut av användningen av LSTM Encoder - Decoder modell som en av våra modeller samt att vi använder högnivå-biblioteket KERAS och Facebooks bibliotek FastText. Förståelsen kan därmed tolkas bristfällande om man enbart kollar på implementationen.

4.1 Word Embedding

För att kategorisera attributen till ett format som datorn förstår kan bland annat *One-hot encoding* användas:

	ärKatt	ärHund	ärFlygplan
Katt	1	0	0
Hund	0	1	0
Flygplan	0	0	1

Denna representation är korrekt men den tar inte hänsyn till “dimensionalitetens förbannelse” vilket är ett problem med större datamängder. Därmed används olika tekniker för att minska dimensionaliteten och förklara mer av den data som finns tillgänglig. Till exempel skulle man istället kunna göra som tabellen nedan. FastText[3] använder denna filosofi för att kunna skapa ordvektorer med information om ordet i förhållande mellan andra ord på ett dimensionssnålt vis.

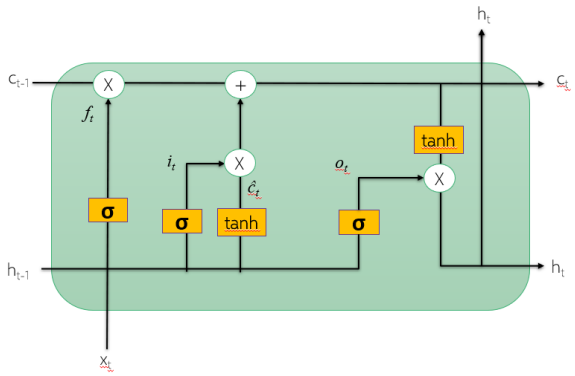
	Djur	Fordon
Katt	1	0
Hund	1	0
Flygplan	0	1

4.2 Long-short term memory[4]

LSTM är en mer utvecklad variant av ett vanligt Recurrent Neural Network (RNN). För RNN uppstår problem med långa sekvenser då den inte kommer ihåg ord från längre tidssteg bakåt. Detta är på grund ut av att de närmaste tidsstegen har för stor påverkan på uppdateringen av det neurala nätverket. Genom att öka minneskomplexiteten med hjälp av fler lager minimeras närmaste tidsstegs påverkan på hela nätverkets vikter. Successivt kommer nätverket ihåg ord från längre tidssteg bort och därmed längre sekvenser. Abstraktionen av en LSTM-cell kan fördelas i tre lager: ett minneslager, ett uppdateringslager och ett outputlager.

[3] "Word vectors for 157 languages · fastText", *Fasttext.cc*, 2020.

[4] Hochreiter, S., & Schmidhuber, J. (1997). LSTM can solve hard long time lag problems. In *Advances in neural information processing systems 9*. Cambridge, MA: MIT Press.



Förklarande figur på Long Short-Term Memory cell där x är ordvektor, c är cellens tillstånd och h är det gömda tillståndet

4.2.1 Minneslager

Minneslagret utgörs av ett neuralt nätverk där det gömda tillståndet och ordvektorn kombineras med vikterna och bias (se funktion nedan). I sigmoid-funktionen blir alla input-värden ett tal mellan 0 och 1 i output. Detta gör att när f_t elementärt multipliceras med c_{t-1} blir vissa index för cell-vektorn 0 och andra blir 1 om vi antar att f_t-vektorn är en vektor med 1: or och 0: or. På så vis kommer c-vektorn ihåg vissa av sina föregående index medan andra glöms bort. Därav namnet, minneslagret.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

4.2.2 Uppdateringslagret

Uppdateringslagret är till för att uppdatera det tidigare cell-tillståndet. För att kunna göra det använder man i_t och $\hat{c}_t$. $\hat{c}_t$ använder sig av tanh-funktionen vilket pressar ner nätverket till en vektor med tal mellan -1 till 1. Detta gör man för att sedan kunna förstora eller förminska i_t:s påverkan på c i outputlagret. Lagret tillför därmed med en utökad förståelse av vad som bör uppdateras eller inte.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\hat{c}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

4.2.3 Outputlagret

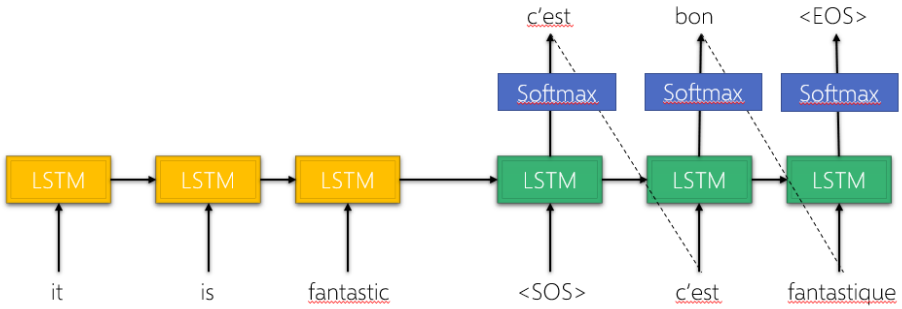
Detta lager är det som sammanställer alla lagerberäkningar och uppdaterar nästa cells tillstånd och nästa gömda tillstånd. Här uppdateras cell-tillståndet av de funktioner som tidigare har beräknats. Därefter kombineras det nya cell-tillståndet med tanh för att beräkna vad som bör förbättras med det gamla gömda tillståndet. Därefter är en iteration av LSTM strukturen gjord och nästa tidssteg påbörjas.

$$c_t = f_t * c_{t-1} + i_t * \hat{c}_t$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * tanh(c_t)$$

4.3 Encoder - Decoder modell



Ovan visas hur en Encoder - Decoder modell fungerar under ett visst antal tidssteg. Modellen fungerar på det viset att kodaren (Encoder) i gult tar in en mening ord för ord i det språket som man vill översätta från. Cellens tillstånd och det gömda tillståndet beräknas för hela cellen. Dessa två tillstånd blir sedan de två initiala tillstånden för avkodaren (Decoder) i grönt.

Anledningen till detta är, på en abstrakt nivå, att all information om meningen tolkas vara samlad i cellens tillstånd och det gömda tillståndet från kodaren. Därmed får avkodaren all den information om meningen som den behöver för att kunna översätta meningen till målspråket. Dock tillförs även meningen på målspråket i avkodaren för att tränas snabbare vilket ofta kallas *Teacher Forcing*. Därefter läggs ett Softmax-lager i koppling till avkodarens gömda tillstånd för att beräkna sannolikheterna för alla möjliga unika målspråksord från korpuset.

Därefter beräknas förlustfunktionen samt att alla vikterna uppdateras genom att använd *Backpropagation Through Time (BTT)*. Till skillnad från en vanlig backpropagation som är en kedjeregel av partiella derivator av föregående neuroner så blir BTTs kedjeregel beroende, utöver föregående neuroner, av tidigare tidssteg i nätverket. Nedan visas skillnaden mellan backpropagation och BTT för en vanlig RNN. Ekvation (1) står för vanligt backpropagation. Dock är n_3 beroende av tidigare n vilket bildar en intern kedjeregel som blir till ekvation (2). För LSTM är detta mer komplext då det finns fler vikt-matriser att uppdatera, vilket underlättar för nätverket att komma ihåg långa sekvenser.

$$\frac{\partial L_3}{\partial w} = \frac{\partial L_3}{\partial \hat{y}_3} \cdot \frac{\partial \hat{y}_3}{\partial n_3} \frac{\partial n_3}{\partial w} \quad (1)$$

$$\frac{\partial L_3}{\partial w} = \sum_{k=0}^{3} \frac{\partial L_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial n_3} \frac{\partial n_3}{\partial n_k} \frac{\partial n_k}{\partial w} \quad (2)$$

4.4 Optimeraren ADAM (Adaptive moment estimation)

ADAM[5] är en förbättring av mini-batch gradient descent. Till skillnad från vanlig stokastisk gradient descent har ADAM en utökad repertoar av parametrar. ADAM har egentligen två delar i sin algoritm, momentum och en adaptiv inlärningsgrad. För att illustrera momentum metaforiskt kan man säga att momentum ter sig likt en boll som rullar ner för en backe; desto brantare backen är desto snabbare rullar bollen. Den adaptiva inlärningsgraden ändras även beroende på hur brant backen är. Däremot ändras denna inlärningsgrad något försiktigt för att den inte ska bli obefintlig och inte konvergera mot minimum av förlustfunktionen.

[5] Diederik P. Kingma and Jimmy Ba, "Adam: A Method for Stochastic Optimization", *arXiv.org*, 2020.

5. Kort beskrivning av implementation

5.1 Bigram-modell

Först testades en implementation med trigram-sannolikheter men det visade sig komma med flera oväntade problem. Till att börja med så krävdes det enormt mycket minne för att skapa nödvändiga matriser, åtminstone med de mängder data som krävs för att motivera trigram-sannolikheter.[6] Även med små mängder data (vilket skulle gett dåliga resultat) blev programmet tungrott så det beslutades att överge trigram till förmån för bigram.

Den metod som valts för att koppla bigram-sannolikheter till översättning är Viterbi-avkodning.[7] Men först följer en beskrivning över hur bigram-sannolikheterna beräknats.

För att beräkna bigram-sannolikheter behövs en träningskorpus. Träningskorpuset består här av texter på målspråket – i vårt fall engelska. Det korpus vi valt att använda är Open American National Corpus[8] som finns att ladda ner gratis på internet och enligt upphovsmännen själva består det av 15 miljoner ord. Texterna är spridda över många genrer vilket bedömdes vara lämpligt för att kunna hantera många olika typer av översättningar. För att minska behovet av ett större träningskorpus rensade vi korpuset från element som var icke önskvärda; siffror och symboler togs bort helt och hållet medan versaler omvandlades till gemener. För bättre resultat kan man göra en mer avancerad rensning av korpuset men för våra enkla tester blev resultaten tillfredsställande. Exempel på att vår rensning kan förbättras är att vi fick “ord” som *himheritthem* som egentligen borde delats upp i *him, her, it, them* men med tillräckligt stor korpus blir dessa fenomen försumbara. Det gäller alltså att hitta en balans mellan storlek på korpus, hur man rensar det och vad som är ett godtagbart testresultat. Något annat man bör fundera över är exempelvis om resultatet blir bättre genom att dela upp korpuset i meningar för att inte få felaktiga bigram-ordföljder när man tar bort en punkt men för vårt ändamål tycktes detta inte vara nödvändigt.

Med vår rensade träningskorpus kunde vi beräkna bigram-sannolikheter i form av logaritmiska tal. Skälen till att använda logaritmiska tal är att dels ta hänsyn till datorns precision genom att inte ha för många nollor som decimaler, och dels att datorn då kan beräkna multiplikation snabbare genom att addera logaritmer. Eftersom många ordföljder aldrig kommer uppstå i korrekt skrivna texter så använder vi *Laplace-Smoothing* för att inte få sannolikheter som är 0. Andra metoder för att hantera nollsannolikheter är backoff och linjär interpolation[9] men backoff ansågs inte lämpligt för bigram eftersom det enbart skulle bli unigram. Linjär interpolation hade varit ett bra alternativ men eftersom Laplace-Smoothing var enklare att implementera valde vi det först med möjlighet att ändra om resultaten ansågs vara för dåliga.

---

[6] Johan Boye, föreläsning “Statistiska språkmodeller” 13/12–2020
[7] Johan Boye, föreläsning “Ordklasstaggning och Hidden Markov Models” 13/12-2020
[8] American National Corpus Project, “The Open American National Corpus” <http://www.anc.org> accessed December 13, 2020
[9] Johan Boye, föreläsning “Statistiska språkmodeller” 13/12–2020

Beräkning av mest sannolika mening sker som sagt genom Viterbi-avkodning. Vår implementation kan tolkas som att man ser på varje enstaka ord i den givna – naiva – översättningen som en *observation*. För varje ord på ursprungsspråket finns det eventuellt flera översättningar på målspråket. De alternativa orden på målspråket bör ses som potentiella kandidater och därmed betraktas som *gömda tillstånd*; ett gömt tillstånd är vad observationen kunde ha varit. En observationssannolikhet fördelas likformigt över observationen och de gömda tillstånden, alla sannolikheter summeras till 1. Den likformiga fördelningen motiveras av att ordval för översättning skett slumpmässigt vi vill med hjälp av bigram-sannolikheterna avgöra vad som är en sannolik ordföljd/kombination av ord. Det går troligtvis att få bättre resultat genom en mer avancerad fördelning av observationssannolikheter men vi valde att förbli trogna enkelheten som vår implementation representerar. Med den likformiga fördelningen blir det enkelt att förstå vilket ord algoritmen väljer: om inget alternativt ord anges kommer observationssannolikheten för det givna ordet att bli 1 och därmed dominerar observationssannolikheten alla bigram-sannolikheter. Om alternativa ord anges kommer det alternativ som har högst bigram-sannolikhet användas.

5.2 LSTM Decoder - Encoder modell

5.2.1 Rensa korpus och initiera dimensioner

Korpuset var en parallellkorpus mellan engelska till franska på 200 000 meningar[10]. Modellen använde sig däremot av enbart 50 000 meningar. Första steget var att rensa skiljetecken och nummer från korpuset. Därefter undersöktes textdatans struktur för respektive språk. Parametrarna var bland annat längden på längsta meningen för båda språk och hur många unika ord som fanns. Dessa dimensioner behövdes för att modellen ska kunna tränas. Samtidigt skapades en python-*dictionary*, för varje unikt ord gavs ett unikt index. Detta för att kunna identifiera för Embedding lagret i modellen vilken ordvektor som ska användas för given input-mening.

5.2.2 Modellens struktur

Själva modellen är uppbyggd av två LSTM lager. Ena LSTM lagret fungerar som en kodare och tar in meningen på engelska, medan det andra LSTM lagret avkodar meningen till franska. Detta görs genom att sekvensen skickas in i kodaren som beräknar dess cell-tillstånd och dess gömda tillstånd. Dessa två tillstånd blir avkodarens initiala värden i kombination med det första ordet för den franska översättningen av meningen. Därefter beräknas cellens tillstånd och det gömda tillståndet för varje ord i sekvensen. För varje tidssteg går det gömda tillståndet igenom ett softmax-lager med dimension motsvarande *antal franska unika ord*. Denna dimension används för att identifiera vilket ord som modellen förutspådde. Ordet med störst sannolikhet anses vara det korrekta ordet.

5.2.3 Träningsmetoder

Träningen underlättas med två metoder. Den första är ett Embedding-lager med förtränade vektorer från FastText[11] för de ord som finns i korpuset. För att simulera en mening för modellen skickas varje ordvektor i ordning och skapar en matris där dimensionerna är unika

[10] "Bilingual Sentence Pairs (From the Corpus Created by the Tatoeba Project)", *Manythings.org*, 2020.
[11] "Word vectors for 157 languages · fastText", *Fasttext.cc*, 2020.

ord i korpuset samt den största meningslängden. Varje sekvens är av samma längd som den längsta meningen i korpuset för att standardisera träningen. Om godtycklig sekvenslängd är mindre än maxlängd tolkas resten av vektorn som utfyllning i form av nollor vilket ignoreras av Embedding-lagren vid beräkning. Dessa förtränade Embedding-lager är kopplat till var sitt LSTM-lager. Den andra metoden för att träningen ska gå snabbare i denna kontext är Teacher Forcing. Där har den korrekta franska meningen förskjutits med ett steg så att modellen lär sig att förutse nästkommande ord utan att behöva förutse hela sekvensen på en gång. Vilket underlättas rent tekniskt med taggarna <SOS> och <EOS> kring målsekvens.

5.2.4 Interferensmodell

När modellen är tränad så byggs två nya modeller med dessa nyträngade vikter som kommer användas för att förutse en hel sekvens av ord. Det man gör då är att enbart använda cell-tillstånd och det gömda tillståndet från den nya kodar-modellen och initierar den nya avkodar-modellens input med <SOS> ordet. Därefter så återupprepar man denna process rekursivt för att förutse en sekvens tills <EOS> taggen förutses vilket och då har man det slutgiltiga svaret.

6. Metod för testning & evaluering

6.1 Bigram-modell

Med den nuvarande implementationen så tar avkodningen för en mening lång tid vilket i stor utsträckning begränsar testmöjligheterna. Det är också svårt att avgöra vad som är en korrekt mening utöver att grammatiken ska stämma då man kan uttrycka samma sak på flera olika sätt. Med dessa saker i åtanke valde vi att med vår kunskap om bigram konstruera noga utvalda testmeningar och se om resultaten överensstämde med våra hypoteser för varje mening.

Som input-meningar valde vi dels grammatiskt inkorrekta meningar som vår modell med enkelhet bör kunna korrigera eftersom bigram-sannolikheterna för grammatiskt inkorrekta ordföljder bör vara väldigt låga. Exempel på en sådan mening är *“and he are”* där *“he are”* inte bör förekomma och bör ersättas med *“he is”*. Vår hypotes är att modellen bör klara alla typer av såna här enkla fel.

Vi prövade också grammatiskt korrekta meningar som dock kan få en annan betydelse om ord byts ut. Exempelvis kan man få att den svenska meningen *“han är en simpel man”* på engelska blir *“he is a gross man”* med hjälp av bab.la eftersom bab.la anger både *simple* och *gross* som en översättning till svenskans simpel. Här är vår hypotes att modellen inte lämpar sig för att göra dessa avväganden eftersom det kommer bero på de träningskorpus som använts. Med den givna svenska meningen vill vi att simpel översätts till simple men även meningen innehållande gross kan vara en tänkbart önskvärd engelsk mening men vår modell kommer alltid ge samma svar för båda. Svaret kommer bero på vilket av orden ”gross” och ”simple” som förekommer oftast i träningskorpuset och inte enbart på relationen till ordet innan.

Slutligen en tredje typ av mening som vi testade var en grammatiskt inkorrekt mening där det inte går att avgöra vilket ord som är grammatiskt korrekt endast med kännedom om det ord som står till vänster. En mening vi valde att prova var *look "how many people there is"* som borde korrigeras till *"look how many people there are"*. Både *"there are"* och *"there is"* är dock tänkbara ordpar i engelskan och vår hypotes är då att modellen kommer välja det ordpar som är vanligast oavsett vad som är korrekt.

6.2 LSTM Encoder-Decoder modell

De två parametrar som användes för att identifiera modellens prestation var träffsäkerhet och förlust. Genom att ha en träningskorpus och ett valideringskorpus fanns det rum för objektiv tolkning hur modellen konvergerade.

Då modellen är relativt känslig av hur man sätter parametrar som batchstorlek, epochs, inlärningsgrad, valideringsgrad och dimension för gömda tillstånd så testades många olika varianter av dessa. Anledningen till denna testning var för att kunna identifiera hur modellen betedde sig vid ändring av dessa parametrar. För hög dimension av gömda tillstånd kan bland annat leda till att modellen blir för anpassad av träningsdatan och kan därmed inte generalisera sig för att anpassas till testdatan.

7. Resultat

7.1 Bigram-modell

Input	Alternativa ord	Förväntat	Output
and he are	*and: {also}, he: {him}, are: {is}*	*and he is*	*and he is*

Svenskans *är* kan översättas till både *is* och *are* på engelska men *he are* är inte grammatiskt korrekt och har därför getts en lägre bigram-sannolikhet. Modellen klarade detta enkla test som förväntat.

Input	Alternativa ord	Förväntat	Output
he is a gross man	*he: {}, is: {are}, a: {an}, gross: {simple}, man: {male}*	*he is a gross/simple man*	*he is a simple man*

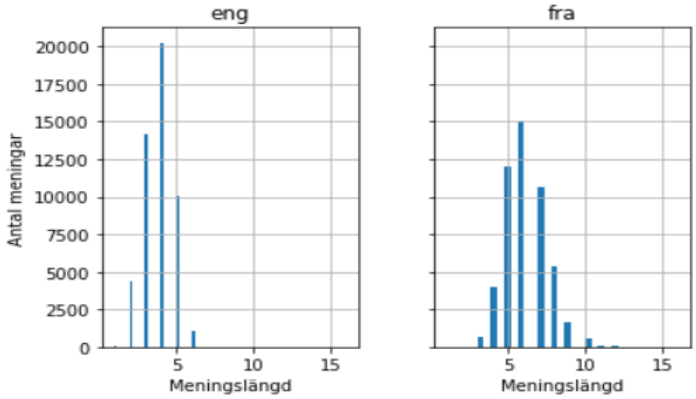
Den erhållna meningen är grammatiskt korrekt men har inte nödvändigtvis förbättrats betydelsemässigt. Det enda, enligt oss, som går att avläsa är att *simple* verkar vara ett vanligare ord och därför mer sannolikt att det är korrekt.

Input	Alternativa ord	Förväntat	Output
look how many people there is	*look: {see}, how: {}, many: {much}, people: {persons}, there: {}, is: {are}*	*look how many people there are*	*see how much people there is*

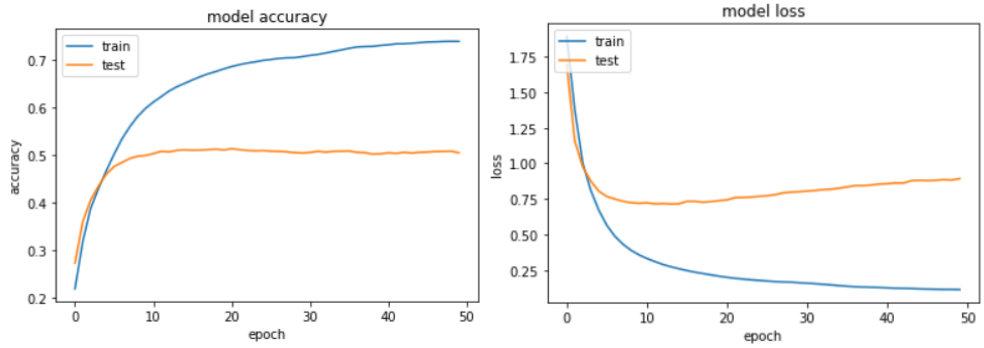
Modellens resulterande mening är sämre än den som angavs som input. Den klarar inte av att justera *is* till *are* givet det som står före *there*.

7.2 LSTM Encoder - Decoder modell

Distributionen av sekvenser är central för att få en modell som kan anses översätta meningar mellan språket som ska översättas och målspråket. Nedan visas disponeringen av sekvenslängd från 50 000 meningar av använt korpus. Den franska delen hade en betydligt högre varians av meningslängden än de engelska meningarna. Antalet unika ord var även drastiskt olika då korpuset hade 5998 unika engelska ord och 12613 franska unika ord.



Dessutom påverkades modellen vid justering av parametrar. Den mest drastiska förändringen av träffsäkerhet och förlust var vid ändring av dimensionen av det gömda tillståndet. Dimensionen som fick bäst träffsäkerhet och minst förlust för valideringen var 128 i dimension. Testning gjordes även för 256 (se figur 1 & figur 2 i Appendix) och 64 (se figur 3 & figur 4 i Appendix). För dimension 256 blev modellen väldigt överträngad vilket resulterade i drastisk kurvskillnad mellan träning och validering för både träffsäkerhet och förlust. Vidare hade dimension 64 svårare att konvergera både för träning och validering vilket resulterade i en låg träffsäkerhet för både träning och validering.



8. Slutsats

8.1 Bigram-modell

De resultat vi erhöll var i hög utsträckning i linje med vår hypotes och våra förväntningar. Det är inte svårt att föreställa sig vilka typer av problem som kan lösas respektive inte lösas med hjälp av bigram. Det enda som förvånade oss resultatmässigt var det tredje exemplet, *look how many people there is*, där resultatet försämrades markant och blev än mer grammatiskt inkorrekt.

Något vi förväntade oss skulle fungera bättre var prestandan. Även med relativt få unika ord (29 000) så var bigram-implementation för långsam för att realistiskt sett kunna motivera de

slätstrukna resultaten. Trigram-modellen hade sannolikt gett bättre resultat med ett större träningskorpus. Problemet med vår implementation av trigram-modellen var att tidskomplexiteten ökar dramatiskt med antalet unika ord. Inte ens med ynkliga 9000 ord lyckades vi få ett resultat inom 20 minuter för en kort mening och därför gav vi upp försöken. Det går givetvis att optimera våra implementationer ytterligare men det hade enbart förbättrat prestandan och inte resultaten. För bättre resultat måste man använda högre ordningar av n-gram men att gå upp i ordningsgrad försämrar den tidsmässiga prestandan markant. Denna prestandaskillnad var tydlig vid vår jämförelse av bigram- och trigrammodellerna.

Sammanfattningsvis kan vi konstatera att resultaten som en n-grammodell ger inte kan motiveras av den dåliga prestandan. Resultaten låg som sagt i linje med vår hypotes och det var också enkelt att implementera men prestandan var undermålig. Vi finner det förståeligt att man inom området dator-översättning gett upp försöken med n-grammodeller till förmån för modernare metoder.

8.2 LSTM Encoder-Decoder modell

Under implementering av modellen samt när resultaten visade sig kunde vi identifiera en relativt bra modell i förhållande till våra tidigare kunskaper om neurala nätverk. Biblioteket KERAS var till stor hjälp och visade sig vara kraftfullt för snabb implementering. Vidare skulle en ökad mängd data vara till stor nytta för att minska överanpassning på träningsdatan. Redan vid 50 procent avstannade konvergensen för valideringen vilket är ett tecken på att modellen inte är bra på att generalisera sin översättningsförmåga för nya data. Samtidigt ökade valideringsförlusten runt 10 epochs vilket även det är ett resultat av överanpassning.

Vidare var den datan vi använde oss av något enformig. Ett flertal meningar hade samma uppsättning vilket kan ha försämrat modellens konvergens mot en bättre träffsäkerhet för både träning och validering. Dessutom var antalet unika ord något få i jämförelse med antal rader vilket inte är att föredra för en översättningsmodell. Modellen kan därmed ha fått en hög bias mot sin träningsdata vilket kan anses något bekräftat i jämförelse med valideringens resultat. Utöver detta fanns ett tydligt mönster i det faktum att spridningen av längden på meningar var större för de franska meningarna än dem engelska. Detta kan mycker väl bero på vårt val av korpus och storlek på tränade rader. Om man däremot antar att fördelningen av längden på sekvenser representerar engelska meningar och dess franska översättning överlag, kan det betyda att engelska som språk är mer informationskompakt än franska. Huruvida detta kan påverka modellens översättningsförmåga är oklart. Att dra någon slutsats kring detta är utanför rapportens analys men värt att undersöka vidare.

Modellen överlag kan definitivt förbättras med mer diversifierat data. Däremot kräver detta troligtvis träningsprestanda som inte var tillgänglig för oss. Dock bör detta vara en något intressant indikation på hur en LSTM Encoder - Decoder modell kan lära sig med relativt få datapunkter. Något som kan vara till hjälp vid översättning mellan språk med relativt små parallella korpus.

9. Referenser

Johan Boye, föreläsning “Statistiska språkmodeller” 13/12–2020

Johan Boye, föreläsning “Ordklasstaggning och Hidden Markov Models” 13/12–2020

American National Corpus Project, “The Open American National Corpus” <http://www.anc.org> accessed December 13, 2020

Johan Boye, föreläsning “Statistiska språkmodeller” 13/12–2020

Diederik P. Kingma and Jimmy Ba, "Adam: A Method for Stochastic Optimization", *arXiv.org*, 2020.

"The remarkable way Google Translate actually works", in *The Independent*, 2020, <https://www.independent.co.uk/life-style/gadgets-and-tech/news/google-translate-how-work-foreign-languages-interpreter-app-search-engine-a8406131.html> [accessed 13 December 2020].

Hochreiter, S., & Schmidhuber, J. (1997). LSTM can solve hard long-time lag problems. In *Advances in neural information processing systems 9*. Cambridge, MA: MIT Press.

10. Appendix

Figur 1.

```
Enter the english sentence: and he are
Enter other words for "and": also
Enter other words for "he": him
Enter other words for "are": is
✔ Completed viterbi decoding in: 327.47
and he is
```

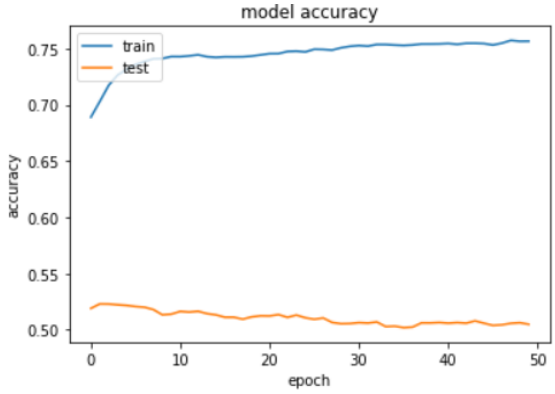
Figur 2.

```
Enter the english sentence: he is a gross man
Enter other words for "he":
Enter other words for "is": are
Enter other words for "a": an
Enter other words for "gross": simple
Enter other words for "man": male
✔ Completed viterbi decoding in: 560.16
he is a simple man
```

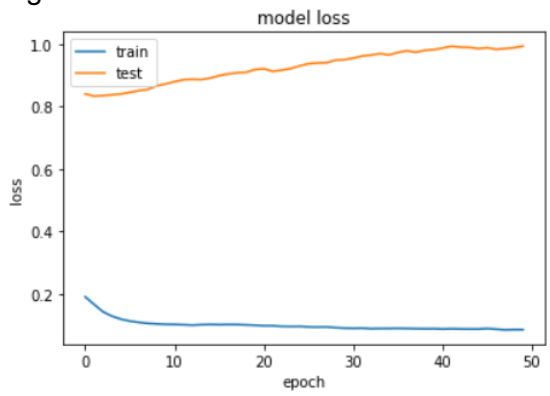
Figur 3.

```
Enter the english sentence: look how many people there is
Enter other words for "look": see
Enter other words for "how":
Enter other words for "many": much
Enter other words for "people": persons
Enter other words for "there":
Enter other words for "is": are
✔ Completed viterbi decoding in: 645.95
see how much people there is
```

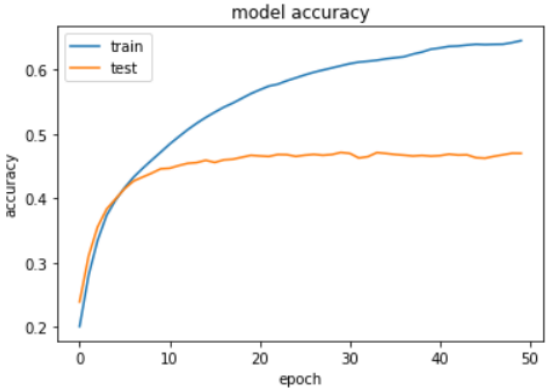
Figur 4.



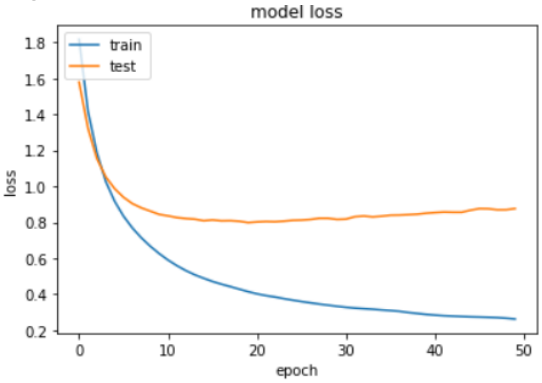
Figur 5.



Figur 6.



Figur 7.



Figur 8.

```
Engelsk mening: ['i', 'hardboiled', 'an', 'egg']
Fransk mening: ['<SOS>', 'jai', 'cuit', 'un', 'œuf', 'dur', '<EOS>']
Förutspådd mening:  jaime une nouvelle carte

Engelsk mening: ['this', 'is', 'boring']
Fransk mening: ['<SOS>', 'cest', 'ennuyeux', '<EOS>']
Förutspådd mening:  cest ennuyeux

Engelsk mening: ['im', 'aware', 'of', 'that']
Fransk mening: ['<SOS>', 'jen', 'suis', 'consciente', '<EOS>']
Förutspådd mening:  jen suis consciente

Engelsk mening: ['ill', 'lend', 'it', 'to', 'you']
Fransk mening: ['<SOS>', 'je', 'vous', 'la', 'prêterai', '<EOS>']
Förutspådd mening:  je te le prêterai

Engelsk mening: ['tom', 'came', 'home']
Fransk mening: ['<SOS>', 'tom', 'est', 'rentré', 'à', 'la', 'maison', '<EOS>']
Förutspådd mening:  tom est rentré chez moi

Engelsk mening: ['youre', 'no', 'saint']
Fransk mening: ['<SOS>', 'vous', 'nêtes', 'pas', 'un', 'saint', '<EOS>']
Förutspådd mening:  tu nes pas un saint

Engelsk mening: ['i', 'cant', 'come', 'now']
Fransk mening: ['<SOS>', 'je', 'ne', 'peux', 'pas', 'venir', 'maintenant', '<EOS>']
Förutspådd mening:  je ne peux pas y aller

Engelsk mening: ['youre', 'annoying', 'me']
Fransk mening: ['<SOS>', 'tu', 'ménerves', '<EOS>']
Förutspådd mening:  tu ménerves

Engelsk mening: ['im', 'blind']
Fransk mening: ['<SOS>', 'je', 'suis', 'aveugle', '<EOS>']
Förutspådd mening:  je suis aveugle
```