AI för dokumentgenerering

Frank Drewes
Institutionen för datavetenskap, Umeå universitet
drewes@cs.umu.se

Kalle Prorok
Institutionen för tillämpad fysik och elektronik, Umeå universitet
kalle.prorok@gmail.com

Sammanfattning

Den här rapporten ger en kortfattad introduktion till metoder och några praktiska resultat från ett AI-texthanteringsprojekt i samarbete mellan Trafikverket, Umeå universitet och Sweco. Tester har gjorts för att extrahera information (geografiska orter, sammanfattningar, frågor och svar) från dokument. Även dokumentgenering, projektets ursprungliga fokus, har adresserats. Där var målet att automatiskt skapa texter för utvalda syften, något som visade sig vara svårt i nuläget då de existerande metoderna är begränsade och samtidigt mycket krävande på datorkraft. Till rapporten hör några förenklade kodexempel där läsaren själv kan testköra och förhoppningsvis lära sig från lite olika fall. Rapporten är indelad i fyra delar: En icke teknisk översikt för allmänt intresserade, en mer detaljerad beskrivning av resultaten, en del om begrepp och metoder för speciellt intresserade samt en del om implementation för programmerare.

Abstract

This report provides a brief introduction to methods and some practical results from an AI text management project in collaboration between the Swedish Transport Administration, Umeå University and Sweco. Tests have been performed to extract information (geographical locations, summaries, questions and answers) from documents. Document generation, the project's original focus, has also been addressed. There, the goal was to automatically create texts for selected purposes, something that proved to be difficult at the moment as the existing methods are limited and at the same time very demanding on computer power. The report includes some simplified code examples where the reader himself can test drive and hopefully learn from slightly different cases. The report is divided into four parts: A non-technical overview for the general interested, a more detailed description of the results, a part about concepts and methods for those particularly interested and a part about implementation for programmers. (Maskinellt översatt via AI/NLP - Google Translate/Machine translated via AI / NLP - Google Translate)

Innehåll

Del I
Introduktion till AI-metoder för texthantering

I projektet *AI för dokumentgenerering* har vi studerat hur artificiell intelligens (AI) kan användas för texthantering inom Trafikverkets verksamhetsområde, dock främst inom området Järnväg. Projektet har fokuserat på metoder för textanalys och dokumentgenerering. Stora delar av arbetet har handlat om stora Artificiella (konstgjorda) neurala nätverk (ANN), ett slags förenklad imitation av det sättet hur man tänker sig att hjärnan jobbar på. På senare år har dessa metoder blivit mycket populära då man kunnat skala upp tidigare försök tack vare ett antal faktorer:

1. Grafikkort som ursprungligen utvecklats för datorspel och datorgrafik är speciellt anpassade för höga beräkningsprestanda och visade sig vara idealiska för den typen av beräkning som krävs för att träna neurala nätverk till ett rimligt pris.

2. För att träna nätverken krävs stora datamängder som numera finns åtkomliga via internet och/eller åtminstone elektroniskt i stora databaser såsom Kungliga Bibliotekets.

3. Nya kraftfulla algoritmer och programvarubibliotek tillhandahålls gratis av stora företag som Google, Facebook och Microsoft, så att de blir åtkomliga för projekt med liten budget.

Inom texthantering används ANN som språkmodeller (se nedan). Mycket av projektarbetet har baserats på de utmärkta svenska språkmodeller som Kungliga Biblioteket (KB) tagit fram [MBH20] utifrån Googles modell Bidirectional Encoder Representations from Transformers (BERT)[1]. KB har unik tillgång till stora mängder text på svenska att träna upp näten på och har en egen github-sida[2].

Projektet har samlat mycket kunskap och erfarenheter och tillämpat dem på några prototyper som vi hoppas ska vara lärorika inför framtiden.

1 Språkmodeller

En språkmodell beskriver hur orden hänger ihop i ett språk. Inom kontexten av det här projektet handlade det mest om svenska. Genom att titta på väldigt många texter kan en ANN lära sig hur meningar är uppbyggda, dvs hur den svenska grammatiken fungerar, men även t ex vilka ord som har samma eller nästan samma innebörd, hur man skiljer mellan olika betydelser av ett och

[1] https://en.wikipedia.org/wiki/BERT_(language_model)
[2] https://github.com/Kungbib/swedish-bert-models

samma ord, vilka ord syftar på vilka andra, och mycket mer. Processen att ”lära sig“ allt det här kallas för träning, och den sker i stort sett genom att parametrarna i ett ANN justeras stegvis i många omgångar. Efter varje steg mäts nätverkets ”förståelse“ av träningstexterna och dess interna parametrar justeras för att förbättra precisionen. Detta upprepas tills man har nått ett slags optimalt tillstånd som inte längre går att förbättra. För att ett justeringssteg inte ska skjuta över målet behöver stegen vara väldigt små och således också väldigt många. I kombination med faktumet att stegen är beräkningsintensiva och behöver ta hänsyn till stora datamängder leder det till att träningen på ett rimligt sätt bara kan genomföras av kraftfulla datorer med de nämnda grafikorten och mycket minne. När en ANN väl har tränats representerar den färdiga modellen förståelse som kräver mycket mindre resurser i sin användning.

Språkmodeller kan skapas och användas på många olika sätt. Historiskt använde man sig inte av ANN alls. För drygt 10 år sedan började nätverken ta över på riktigt, först med förhållandevis enkla nätverksstrukturer (så kallade arkitekturer), sedan med allt komplexare arkitekturer vars träning kräver något som på engelska heter *deep learning*. Vårt projekt har fokuserat på den för närvarande senaste arkitekturen, den så kallade transformern[3].

2 Användningsområden

Vad har man då för nytta av en språkmodell? Det är lite olika men vitsen är att man ska ha nytta av att man vet hur orden hänger ihop och används. Dels enstaka ord men även i form av meningar och då speciellt i de mer avancerade formerna såsom BERT som baseras på den redan nämnda transformern. Transformer-tekniken är en speciell form av ANN som har visat sig vara väldigt effektiv. Man kan även träna den på flera språk samtidigt och då få en flerspråkig (multilingual) språkmodell. En av de stora fördelarna med BERT och liknande språkmodeller är att de är förtränade. Det stora jobbet har alltså redan gjorts av andra med tillgång till de resurser som krävs (t ex Google eller, när det gäller den svenska versionen, Kungliga biblioteket). Sedan kan man antingen använda sig av modellen som den är eller finjustera den för speciella ändamål, t ex när dokument som använder sig av speciell terminologi inom järnväg ska bearbetas.

Exempel på saker man kan göra med den färdiga modellen efter träning är:

2.1 Klassificering av text

Här försöker man känna igen om texten (t ex en recension) har en viss egenskap. Man kan t ex vilja veta vad texten handlar om (t ex ifall den handlar om järnvägs- eller biltrafik), om den uttalar sig positivt eller negativt om något (t ex om man vill förhandssortera inkommande reaktioner från allmänheten på järnvägsplaner). En välkänd tillämpning som inte är begränsad till Trafikverkets verksamhetsområde är spamfilter för epost, där nätverket känner igen och filtrerar ut oönskade meddelanden.

[3] https://en.wikipedia.org/wiki/Transformer_(machine_learning_model)

2.2 Sammanfattning av text

Genom att automatiskt sammanfatta dokument underlättar man för läsaren att få en översiktlig hyfsad förståelse för vad texten handlar om. Språkmodellen identifierar då centrala meningar och begrepp i dokumentet och skapar en kort text utifrån det. Med andra ord, algoritmerna plockar ut meningar från texten som verkar viktiga. Detta är något vi har gjort i vårt projekt. Mer avancerade tekniker försöker formulera om texten och eventuellt även skapa ny text som beskriver lite mer översiktligt och på ett bättre sätt än enstaka utvalda meningar vad dokumentet handlar om.

2.3 Hitta ställen där önskad information finns

Språkmodeller kan också med fördel användas för att känna igen om ord betecknar platser, personer, mått, tidsangivelser, organisationer, en lag eller vad man nu tycker är viktigt. Arbetsförmedlingen gjorde t ex en modell som kände igen kompetenskrav i jobbannonser för att matcha ihop det med arbetssökandens beskrivningar. I samband med beskrivningar av järnvägsprojekt kan det användas för att få reda på vilka orter som berörs, om hotade djur- eller växtarter nämns, och dylikt.

2.4 Översättning

För att automatiska översättningar ska bli bra behövs språkmodeller. Transformerarkitekturen är speciellt bra lämpad för det. Mycket förenklat uttryckt konstruerar nätverket den översatta meningen ord för ord genom att leta upp de ställen i originalmeningen som brukar vara mest avgörande för översättningen av näst kommande ord. Den kollar sedan upp dessa ställen för att kunna avgöra vilket ord vilket eller vilka ord passar bäst. För att kunna göra det behöver modellen ha lärt sig språkets struktur, dvs hur orden i en mening hänger ihop. Om t ex *the game fled into the forest* ska översättas till svenska är översättningen av *the game* inte *spelet* utan *viltet*, men för att förstå det behöver modellen ha lärt sig att spel inte flyr in i skogen.[4]

2.5 Frågor och svar på text

En mycket mycket intressant och potentiellt värdefull tillämpning av språkmodeller är frågor och svar: Modellen matas med en text och sedan kan användaren ställa frågor som modellen besvarar genom att plocka fram motsvarande information från texten (förutsatt att texten innehåller svaret). Man behöver alltså inte själv leta i långa dokument som t ex miljökonsekvensbeskrivningar utan får omedelbart svaret man letar efter.

[4] Intressant i sammanhanget är att Google Translate i skrivande stund inte klarar av det här exemplet när man översätter från engelska till svenska. Däremot fungerar översättningen från engelska till tyska, vilket indikerar att modellen för översättning till tyska är mer sofistikerad. Även den slutar dock fungera när meningen kortas ner till *the game fled*, vilket antyder att inte ens översättningen till tyska tar hänsyn till *fled* utan lägger fokus på ordet *forest*.

Tyvärr saknas än så länge bra svenska träningsmängder för att modellerna ska bli riktigt bra på frågor och svar, men projektet SuperLim som slutfördes i Juni 2021 kom fram till vissa användbara testdata för att t ex se om meningar hade snarlik betydelse. För närvarande använder man översatta utländska exempelpar på frågor och svar för att träna nätverken, vilket resulterar i en förmåga på ca 70 % korrekta svar (om man räknar snällt) när det gäller svenska. Med en hyfsat stor svensk träningsmängd borde man kunna skapa tillförlitligare modeller.

Exempel på fråge och svar-system är Stanford Question Answering Dataset (SQuAD)[5] , General Language Understanding Evaluation (GLUE) benchmark[6] för engelska och det ryska DeepPavlov[7] som även hanterar generella språk via Googles flerspråkiga (Multilinguella) BERT.

3 Språkmodellernas struktur och spridning

Språkmodellerna är ofta gjorda och hanteras i speciella programbibliotek, förr var Googles Tensorflow/Keras vanligast eftersom det är enkelt att lära/hantera men Facebook/Microsofts PyTorch har blivit en allt populärare uppstickare som är lite mer flexibel men samtidigt lite krångligare då man behöver göra en del saker (ange derivering och kopiering) själv. Huggingface är en populärt sajt för att lagra sina tränade språkmodeller på så att andra kan använda dem (oftast gratis och med båda typerna av programbibliotek och i flera varianter med t ex bara små bokstäver eller både stora och små).

4 Lite kort om projektets resultat

En del av projekttiden användes för att sätta sig in i aktuell status forskningsmässigt, det sker oerhört mycket nästan varje dag, mest utomlands men även i Sverige. Det har varit inläsning på vetenskapliga artiklar ca en timme varje morgon samt deltagande i föredrag och möten 1-2 gånger per vecka under över ett års tid så även om det inte är jättesvårt så är det rätt mycket nytt att lära sig, även för erfarna personer och det ändras dessutom ständigt så det kan vara svårt att hålla sig uppdaterad och få en bra översikt. Vi hoppas att denna rapport bidrar till att underlätta det för andra. Sedan har tid ägnats åt experiment, i den här branchen handlar det mest om att programmera, felsöka och testköra med olika former av indata och justera parametrar så att resultatet blir någotsånär bra. Eftersom det är ett väldigt nytt forskningsområde finns ännu inte så många exakta statistiska utvärderingsmetoder utan man får nöja sig med ungefärliga prestandasiffor om hur många vettiga svar man får och bedöma kvaliteten på sammanfattningar m m.

[5] https://rajpurkar.github.io/SQuAD-explorer/
[6] https://gluebenchmark.com/
[7] https://deeppavlov.ai/

4.1 De första experimenten

Till en början provades olika varianter av språkmodeller för att extrahera information från Miljökonsekvensbeskrivningar som hämtades hem från Trafikverkets öppna webbsidor för Norrbottniabanan[8]. Vi kunde automatiskt få fram listor på inblandade orter och personer samt svar på enklare frågor och sammanfattningar av dokumenten i olika detaljnivåer. Resultatet presenterades för Trafikverkets personal i ett seminarium som lockade över 300 deltagare och det gjordes en Välkomstfilm[9] och det finns även OH-bilder[10]. Många (ca 30) av dem hörde av sig för vidareutveckling av idéer som uppkommit. Tack till alla som engagerat sig! Dock har vi av GDPR-skäl valt att inte publicera namn på kontaktpersoner etc.

4.2 Några prototypsystem

Vi konstruerade små prototypsystem för att testa några av de uppkomna idéerna. Andra var väldigt intressanta men utanför vårt område, t ex att scanna av vägkvalitet, se skyltar lite snett från fordon, genomföra röst- och ljudanalys och liknande AI-relaterade problem. Frågor som dessa borde kunna hanteras av liknande projekt fast med motsvarande inriktning.

Vi pratade också om några intressanta projekt som det sedan ändå inte blev någon prototyp av, t ex verkar Diariehantering lovande, man skulle kunna klassificera typ av dokument automatiskt, göra en sammanfattning och hitta lämpliga ord att tagga dokumentet med för att snabbare kunna hitta och ögna igenom viktiga delar av dokumentet men hanteringen består ju av många andra saker också och det är säkerhets- och lagliga aspekter inblandade.

Vi testade att hantera information via Trafikverkets öppna webb-api:er[11], där kan man t ex kan läsa av trafiksituationer och förseningar direkt via dataprogram och hantera dem på lämpligt sätt, t ex försöka automatiskt skapa lämplig text för presentation ute på perrongerna så att blivande passagerare blir informerade eller snabbt göra lämpliga åtgärder såsom att tillkalla jägare för att få bort djur, hantera vädersituationer med snöskottare eller tillkalla poliser för att avhysa personer på spåren. Att generera längre texter provades via en extra kraftfull språkmodell men tyvärr visade resultaten att tiden inte är riktigt mogen för det ännu, i alla fall inte på svenska. Flera lovande försök är dock på gång så att man redan lite senare på hösten 2021 kan förvänta sig att det kommer att bli lättare.

En populär tillämpning av språkmodeller där vårt fråge/svar-system i en vidareutveckling borde vara användbart är chat-robotar. Dagens chat-bottar jobbar till stor del med en mängd förprogrammerade fraser. Med vår teknik kan man däremot undvika mycket av det tröstlösa arbetet och låta datorn hitta formuleringarna själv. Det undviker inte bara en hel massa manuell programmering

[8] https://www.trafikverket.se/nara-dig/projekt-i-flera-lan/Norrbotniabanan/
[9] https://www.youtube.com/watch?v=oSAiTDXbOho
[10] https://github.com/nutte2/TRFV/blob/main/AI_NLP_TRFV_UMU_2_TS%20(3).pdf
[11] https://api.trafikinfo.trafikverket.se/

utan gör också systemens beteende naturligare i användarens ögon.

Trafikverket har stora mängder dokument och vi gjorde ett försök att hitta referenser i dokument till andra dokument för att se om man på så sätt automatiskt skulle kunna få en överblick över vilka dokument som hänvisar till vilka. Detta skulle bl a kunna användas för att spåra dokument som behöver uppdateras när ett som hänvisas till ändras. Det har nyligen (juni 2021) skapats databaser med en massa textexempel. Tack vare dessa kan man träna upp språkmodeller för svensk text som kan hitta motsägelser, synonymer och annat i stora dokumentsamlingar. Tyvärr kom dessa databaser för sent för att kunna genomföra sådana experiment inom projektet. Istället lämnade vi in forskningsansökningar för fortsättningsprojekt relaterade till våra prototyper, men än så länge utan framgång. Det skulle finnas mycket tid och pengar att spara bara man får de här teknikerna att fungera, men samtidigt innebär sådana projekt stora utmaningar och det är inte garanterat att resultaten kommer att bli bra.

4.3 Effektiviserade vattenanmälningar

Det vi tittat mest på i slutet av projektet är att hantera Trafikverkets ansökningar mot länsstyrelserna på ett effektivare sätt; för varje litet vattendrag som man vill bygga bro över så behöver en så kallad vattenanmälan att göras. I Västerbotten[12] handlar det om att fylla i ett webbformulär på sju sidor med faktasiffor, kryssrutor och små kartor och bilder som kan hämtas från olika dokument och sammanställas av ett dataprogram. Vi tänker oss att automatisera ifyllandet av formuläret baserat på en beskrivningsfil, en textartad fil som talar om vilka data som ska hämtas/sökas var och som relativt lätt kan anpassas för andra användningsområden, när formuläret ändras eller för andra länsstyrelser.

4.4 Närmaste framtiden

Senare under hösten hoppas vi få fram en webb-server där Trafikverkets personal och andra intresserade kan provköra några av de saker vi konstruerat. Planen var att få fram den tidigare, men långa leveranstider för den nödvändiga hårdvaran fördröjde arbetet. Servern kommer att kunna köras under begränsad tid eftersom hårdvaran kan behövas till andra, framtida projekt, troligen kommer det bli halva dagtiden under November 2021. Oss veterligt blir det då den enda maskinen i Sverige av sitt slag, superdatorcentrumen har kraftfulla datorer men arbetar i batch-form (ett jobb i taget i turordning) så väntetiden kan vara flera veckor, inte så bra för ett fråge-svar-system..och många andra öppna system har inte tillräckligt kraftfulla beräkningsgrafikkort för att hantera stora språkmodeller snabbt och de behöver ofta ladda hem modellerrna på begäran och redan det tar många sekunder medans våran dator redan har de önskade modellerna hämtade och inlagda, färdiga för bruk. Lite mer långsiktigt kommer exempelkod att öppet göras tillgängligt för att kunna testköras lite mer om-

[12] hhttps://www.lansstyrelsen.se/vasterbotten/miljo-och-vatten/atgarder-och-verksamheter-i-vatten/vattenverksamhet/anmalan-om-vattenverksamhet.html

ständigt och långsamt på egen maskin eller på en speciell hemsida, t ex Googles Colab, efter att testtiden gått ut.

4.5 Nytta

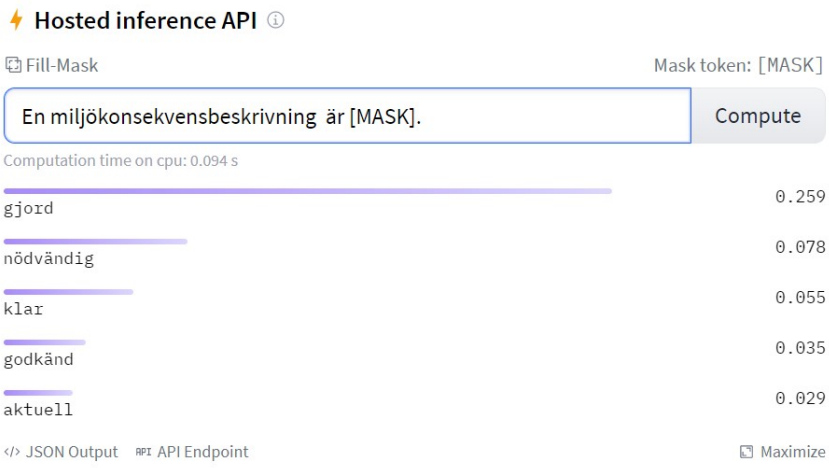
Vad har/får man då för nytta av sådana här Natural Language Processing (NLP)-system? Redan idag används de mycket, t ex Google translate, även om det inte är perfekt så underlättar det översättningar och ger förståelse mellan olika länder/språk. Google använder det för sökningar så man numera kan söka på hela meningar istället för enstaka ord. Apples Siri, Amazons Alexa, Microsofts Cortana och Googles Assistent är exempel på röststyrning som kan vara utmärkt om man t ex framför ett fordon eller har händerna upptagna. I vårt projekt har vi visat hur man snabbare/bättre/enklare/billigare kan hitta, organisera, sammanfatta och delge (textbaserad)information, t ex för att hålla en högre service på meddelanden på perronger, få struktur i stora dokumentsamlingar, hålla spåren fria och hjälp med att fylla i krångliga formulär samtidigt som man sparar pengar och tid. Inom några år kommer AI/NLP-tjänsterna troligen vara mer allmänt användbara och rentav självklara och rimligen möjliga att köra på billigare hårdvara.

Del II
Mer om projekts resultat

5 Inledande demo/försök

5.1 Inledande tester på planbeskrivningen 180704

För att se hur väl algoritmerna skulle klara planbeskrivningar testade vi KB-BERT initialt med några typiska uppgifter inom språkhantering för att sedan applicera det på en riktig planbeskrivning. Ett litet exempel på de allra första testerna (inte vår server) visas i Figur 1.



Figur 1: Ett exempel på hur BERT gissar på det saknade ordet

Efter några lovande resultat som detta testade vi algoritmerna på dokumentet `180704_planbeskrivning.pdf`. Uppgiften vi började med var att identifiera orter, platser, personer angivna i dokumentet. Typiska resultat såg ut som nedan i en preliminärversion med manuellt grovrensade sidor från sidrubriker med ibland avbrott mitt i meningar pga gränsen på 512 delord:

Platser:
Norrbotniabanan, Umeå, Skellefteå, Dåva, godsbangård, industriområde, E, Tavelsjöleden, Skjutfältsvägen, Fäbodvägen, Storraningsbäcken, Fäbod, Ersmarksberg, Ersmark, Ersboda, Tavelån, Gamla, Ersbodavägen, Fällmyran, Sundbäck, Järnväg, Natura, 2000Järnvägen, Ersmarksbergets, Hömyrtjärnsbäcken, Hjåggmarksbäcken, Botniabanan, Norrlandskusten, Luleå, Piteå, Norrland, Stambanan, Norrlands, Robertsfors, Ostvik, Norrbniabananorrbotniabanan, C, NorrlandStambanan, Bräcke, Haparandabanan, Malmbanan, Ådalsbanan, Mittbanan, Norra, stambanan, Botniabananotniabanan, Sverige, Nyland, Kramfors, Härnösand, Timrå, Sundsvall, E12, Östra, skjutfältsvägen, universitetssjukhus, Nordic, Logistic, Center, Norrtågssbang, Holmsund, kommunUmeå, kommun, Vännäs, Bjurholm, Vindeln, Nordmaling, Umeåban, Dåvamyran, Ume, älv, Stadslid, skjut, stadamba, Väg, E4, Ersängsskolan, Hamptjärnsberg, Fäbodberget, Vadforsens, SundbäckJärnvägen, Umeåang, Västra, Ersbodas, Västerbotten, Umeälven, Kustlandsvägen, Grisbacka, Täfteån, Vadforsen, Sveriges, KoliärJärnvägen, Ersmarksvägen, kommuns, Fungnäsmyran, 2000Ersmarksberget, Holmbäcksnäsforsen, Åkrok, Ersbodavägenbar, Vilmyran, Holmbäcknäsfors, Åkroken, Tavelåns, Nyåkersforsenyåkersfors, Nyåkersforsen, Stor, -, Hjåggmarksbäckenjåggmarksbäcken, Åkerholmen, Täftebölesjön, Kullabäcken, Tavelsjö, länken, Ersforsen, meandernäs, koloni, Sävar, Ersparken, I, Tjälamark, Tavelsjöberget, Nyåkersberget, Björsängesberget, stad, Tvärån, Bottenviken, Sundsbäcken, Djupbäcken, Sandbäcken, Holmbäcken, Vindelälvsåsen, gamla, Ersmarksberget, godsban, ErsmarksbergP, Bullmark, Degermyran, Ersbodava, Erstunn, Bergskärningen, räddningsplatsenpå, Nya, Fäbovägen, väg, Fäbodvägens, Traktorväg, Ersmarkstunn, Vännäsbanan, Stad, ByggnadsverkF, Vägbron, Hamptjärnsstugan, Vallen, industriområdebod, För, åkerholmen, JärnvägJärnvägen, RennäringJärnvägen, YresurserJärnvägen, Närhet, Godsbangård, Hömyrtjärnbäcken, Täftebölesjöns, industriområdeJärnvägen, Västerbottens, län, ErsbodavägenP, Norrbotniabana, Ersbodaorrbotniabanan, terminal, ErsbodaJärnvägen, JärnvägJärn, UmeåJärnvägen, Umeåme, TavelånGrundvatten, socken, Tidplan, Gris,

Personer:
Ran, sam, Forslund, norn, nornvi, åkerholm, Olle, Rans, Berg, åker, Di, Bengt, Viktors, vall, klaras, Er, Tor, Tobias, Flygar, Rena, Natur,

Organisationer:
Länsstyrelsen, Försvarsmakten, LänsstyrelsenStamba, kommunal, Tyréns, AB, Umeå, universitet, UmeåUmeås, Forslunda, vattenverk, vattenverket, Kraftvärmeverket, länsstyrelsen, koloniträdgårdsförening, Föreningen, Kulturbrobyggarna, WHO, Naturvårdsverket, MIF, NBB, Markavvattningsföretag, Naturvårdsverkets, författningssamling, kommunalJ, Mark, -, och, miljödomstolen, tingsrätt, Länsstyrelsens, Lantmäterimyndigheten, miljööverdomstolen, Vägverket, lans,

Mått:
ca, 12, km, 1, , , 6, lång, 966, 000, m2, 734, 303, 32, 500, 4, 922, 655, miljarder, svenska, kronor, upp, till, 30, procent, 1000, ton, 1600, mellan, 200, -, 250, dygn, 740, över, 70, 15, %, 25, 3, m, bred, 55, km2, 360, 20, 140, 2, meter, 0, 65, m3, gånger, per, år, 100, Mkr, 150, 10, djup, 5, 8, 7, hög, 790, tidvis, 1800, cirka, 50, meterJärn, 600, 550, 420, 380, m33, inom, 80, 677, 750, 57, 29, 711, 298, 880, 14, 800, 7200, 34,

Tider:
samtidigt, under, drifttiden, byggtiden, idag, byggnadstiden, till, 2018, fem, år, 2016, års, lång, tid, i, januari, 2013, Tidigare, per, dygn, framtiden, årsskiftet, Under, de, senaste, åren, 1965, dag, 16, årvärme, 2011, -, 08, 29, 2009, 03, 19, 1981, 24, 1997, 05, 2005, 02, dagen, över, sommaren, hösten, 2017, 1600, talet, historisk, medeltiden, för, 3000, sedan, alla, tider, vintertid, sommartid, från, 1923, vid, denna, idagÅ, slutet, av, 1700, 1899, 1917, Redan, Strax, vinterhalvåret, skidsäsongen, Idag, nyligen, vinter, 2027, 1892, 1953, Ibland, vintern, mellan, 1964, och, 1974, två, tillfällen, ett, tillfälle, februari, mars, våren, 120, den, tiden, tidigare, perioden, augusti, [UNK], 202, på, sikt, 10, 000, dagsläget, redan, Samtidigt, inom, kort, april, juli, juni, byggsked, vart, femte, 2012, 60, månader, omedelbart, hur, 1998, god,

Arbeten:
miljöbalken, Svensk, Standard, Luftguiden, Gamla, Enligt, Skogsvårdsstyrelsens, kartdata, Järnväg, JU, JordbrukP, N, MSBFS, Ekonomiska, målJärnvägs, Umeå, Kulturmiljölagen, och, SKAPA, [UNK], Night, no, guide, Europe,

Sedan testade vi sammanfattning av de ca 200 A3-sidorna ner till ca en sida och det var av typen välj ut meningar som verkar viktiga och kan ses i nedanstående textrutor. Detta kördes under Windows, med en bättre pdf-omvandlare såsom t ex Tika under Linux blir nog resultaten renare.

Sammanfattning, del 1 av 2:
- Samspelande arbets-, utbildnings- och bostadsmarknader genom regionförstoring. Bron utformas så att människor och medelstora däggdjur kan färdas under den. Konsekvenserna avseende funktion och samhälle bedöms sammantaget som positiva. Mycket negativa konsekvenser bedöms uppstå med hänsyn till masshantering. För luft och klimat bedöms positiva konsekvenser uppstå. Den mark som tas i anspråk är belägen inom Rans samebys vinterbetesmarker. I Umeå ansluter banan till Stambanan genom övre Norrland. Sundsvall i söder till Luleå i norr, stora transportbehov. Drumlinerna är lokaliserade i isens rörelseriktning. Kring Figur 3.6-1 Kulturmiljö - strandnivå för 3000 år sedan. På odlingsmarkerna gårdar med tillhörande ekonomibyggnader. Koloniområdet Vadforsen är beläget på den norra sidan om Tavelån. En fågelinventering har genomförts från väg 364 och fram till Hjåggmarksbäcken. Området utgör livsmiljö för ett antal skogsanknutna fågelarter. Mellan Vadforsens koloniområde granskog med höga naturvärden. Järnvägen passerar genom skog och hyggen utan högre naturvärden. Åkroken och Holmbäcknäsforsen Åkroken är namnet på en av Tavelåns meanderbågar. Området påverkas främst av buller från Gamla Ersbodavägen. genomförande av skyddsåtgärder mot höga buller- och vibrationsnivåer. Beräkningarna indikerar relativt låga halter i hela utbredningsområdet. vägområden är halterna låga och där uppfylls miljömålen. Miljökvalitetsnormer och miljömål2) och partiklar (PM10 och PM2,5). Rans sameby har därför tidigare undvikit att nyttja markerna öster om väg E4. Djup till berg varierar mellan 0 m till ca 19 m under markytan. Grundvattenytan i myrområdena mot slutet av sträckan är påverkad av dikning. sediment av silt och lera med varierande mäktighet. Utöver dessa bergarter förekommer även en grovporfyrisk granitoid.

Sammanfattning, del 2 av 2:
Tre av cellerna är avslutade och innan den är full. På deponins norra del har det deponerats industriavfall. Detta främst från närsalter, men även från tungmetaller. I anslutning till detta anläggs tryckbankar som stabilitetshöjande åtgärd. Normal förstärkning består i detta fall av bergbultar och sprutbetong. Eftersom diket skärs av kommer ett nytt utlopp att anläggas. dagsläget breda ut sig långt från sin normala fåra. Slutlig omfattning på bergtekniska åtgärder kommer att bestämmas i byggskedet. Tunneln kommer att tätas för att förhindra att vatten läcker in. Erosionsskydd för diken bör alltid utformas med bergkross. Vägen kommer att sänkas med ca 2,5 m för att möjliggöra passage. I planen redovisas förslag på placering och utformning av enskilda vägar. Nya Fäbovägen kommer att fungera som både service- och ersättningsväg. Fäbodvägens passage med järnvägen (km ca 2+500) kommer att stängas. Söder om pumpstationen vid passagen anläggs en vändplats för servicefordon. Dessa placeras vid västra och östra tunnelpåslaget. Detta för att minimera påverkan på det småskaliga mosaiklandskapet vid Tavelån. För dessa överskrids riktvärdet vid fasad trots åtgärder. Järnvägen bedöms vara positiv för lokalsamhälle och regional utveckling. Järnvägen avskärmas från Ersmark av ett skogsparti. Öster om väg 364 går järnvägen till övervägande del på bank i skogsmark. Detta medför att markanspråket blir förhållandevis stort. Företagens transportkostnader bedöms minska med upp till 30 procent. Buller och vibrationer från entreprenadmaskiner, sprängningsarbeten, mm. Måluppfyllelsen bedöms med hänsyn till detta som god. Vid lokalisering av järnvägen har antalet passager över Tavelån reducerats. I projektet säkerställs att vandringshinder inte uppstår i vattendragen. Målet motverkas såväl permanent som under byggskedet. Järnvägen kommer att medföra permanenta grundvattensänkningar. Konsekvenserna för grundvatten som resurs är således begränsad. Inga våtmarker med högre värden kommer att påverkas av projektet. Måluppfyllelseattraktiva boendemiljön och omgivningen. Tabell 6.5-2 Samlad bedömning av projektets konsekvenser för miljön. Vägrätten upphör när vägen dras in från allmänt underhåll. Hantering av massor kommer att sammanfattas i en masshanteringsplan. Deponering ska endast ske om ingen annan användning är möjlig. Stadsplan för Ersbodaområdet, nordvästra delen, antagen 1981-08-24. När anläggningsarbetena är färdiga genomförs BEST-arbeten. Ersättning betalas till fastighetsägaren för den tillfälliga nyttjanderätten. Ersättningen bestäms utifrån reglerna i expropriationslagen. URL: http://www.miljomal.se/Miljomalen/ Naturvårdsverket.

Genom att anropa Google Translate inifrån programmet kunde man skapa översättningar av resultat, t ex sammanfattningar, kan ju vara lämpligt att t ex sammanfatta även på finska i Norrbotten. Tyvärr stöds inte Samiska av Google translate för närvarande men det finns en norsk sajt som verkar göra det.

Frågor och svar på långa texter kan göras på några olika sätt, dels med en tidig variant av KB-BERT med frågemöjlighet men även med den ryska programvaran DeepPavlov och Googles mångspråkiga BERT. Se listan nedan för några exempel. Det här är ju en tidig version som inte är så imponerande men det sägs komma en svensk version av fråge/svars-exempelbanken GLUE hösten 2021 och då kommer det bli lättare att träna upp näten så de hanterar frågor och svar bättre.

Nedan följer exempel på frågor och svar med en variant av DeepPavlov (många fler resultat skrevs ut, så det här är bara ett utdrag). Resultaten är ofta inte särskilt relevanta, men bedömningen borde återigen göras i ljuset av att varje block på 512 delord behandlas separat, så att block som inte innehåller relevant information leder till dåliga svar.

```
Var sträcker sig stambanan?

från Bräcke till Luleå
mellan 200-250 km/h
mellan Umeå och Skellefteå
12 km
mellan Umeå, Skellefteå, Piteå och Luleå
genom övre Norrland
mellan Umeå och Luleå
km 0+000 till km 12+423 i spårets längdmätning
genomgående linjeföring
Stambanan genom övre Norrland sträcker sig i inlandet från Bräcke till Luleå
norra och södra Sverige
övre Norrland
mellan Ersboda och Ersmark
väg och järnväg
övre Norrland
strax norr
mellan Ersboda och Ersmark
norra delen av en detaljplan
Järnvägsplanen tangerar detaljplanelagd mark vid Dåva industriområde
genom övre Norrland
nord-sydlig
Järnvägen passerar mellan två stadsdelar i Umeå kommun
mellan Ersmark och östra
Järnvägen börjar i utkanten av Umeå godsbangård
ca 3000 år sedan
sträckte sig genom Umeå och vidare mot Täfteå och norrut
Järnvägen går genom I 20-skogen
Tavelån meandrar genom området
väg 364
15-25 m bred
364 och Gamla Ersbodavägen
Järnvägen passerar genom skog och hyggen utan högre naturvärden
väg 364 och första passagen av Tavelån
norrut
Ersmark-Anumark
364
två större bostadsområden
364
från norr till söder
öster om väg 645
dels åt norr mot Tavelån och dels åt söder
Tavelån har ett slingrande lopp och ett antal järnvägspassager
högre partier utgörs terrängen av drumliner och små moränryggar
ca 0-1,5 m
0+000-2+750 808441.25 öst-väst till sydväst-nordöst
från norr till söder
södra delen av deponin
väg 364 och väg 645
Järnvägen kommer däremot att passera ett dike
12 km
Umeå-Dåva
nära anslutning till koloniområde och bostadsmiljöer i öppet landskap
0+000-2+750
1,6 km lång
Ersmark och Ersboda
```

strax norr om elljusspår i Ersboda och Vadforsens koloniområde
ca 11+150
8,2 m
5,5 m
halva spårtunnelns längd
395 m
Mellan tunneln och mötesstationen vid Ersboda (km ca 6+000)
tre ställen
12 m eller mer
längs med tunneln
km ca 6+900-7+800
Vid km ca 0+400

Vad innebär bristande kapacitet?

Att banan är enkelspårig
Att banan är enkelspårig
Järnvägsplanen
grundvatten
17 kap miljöbalken
Nollplus-alternativ
en ny bana längs kusten
timerad linje
Att banan är enkelspårig
Kraftvärmeverket
igenväxande diken
vägtrafikbuller
igenväxande marker
ster
vattnet blir surt
försurat vatten
erget begränsad hållfasthet
järnvägssträckning
Alternativet
Linjerna
alternativ
tryckbankar
tryckbankar
tunnelsäkerhet
Planerad järnväg
järnväg
mindre kostsamma metoder
linjestudien, där den sträckning som innebär minst påverkan på miljön
bullerskyddsplank
driftskedet
Järnvägssträckningen stängslas
järnvägen
Järnvägsplanen
Luft Järnvägen
Rennäring Järnvägen
ytvattenresurser
överskott
fler tillgängliga spår som kan trafikeras vid händelse av en olycka
planalternativet
Norrbotniabanan

```
re tåg
byggnadstid
Brandfarliga och explosiva varor
Norrbotniabanan
utsläpp
```

En ny körning med en variant av KB:s version av BERT (alltså inte DeepPavlov) försök gav i viss mån bättre svar; vi använde tröskelvärde $0,1$ på denna experimentella modell:

```
qa = pipeline('question-answering',
model='KB/bert-base-swedish-cased-squad-experimental')
```

vilket resulterade i följande svar:

```
Var sträcker sig stambanan?

inlandet från Bräcke till Luleå
övre Norrland
övre Norrland
övre Norrland
övre Norrland,
I 20-skogen
Fäbod- och Ersmarksberget.
från Umeå godsbangård
20-skogen
Där järnvägen går på bank
Tavelån,
Tavelån.
Norrbotniabanan
I 20-skogen
Tavelån
mellan Umeå och Skellefteå
Sävar.
Sävar.
Tavelån.
Robertsfors.

-------

Vad är största tillåtna axellast?

25 ton
25 ton

-------

Var kommer det finnas stationer och var kommer resandeutbyte att ske?

Dåva industriområde
Umeå C och Umeå Östra
Dåva industriområde.
Ersboda
Ersboda

-------
```

```
Vad kostar projektet?

1,655 miljarder svenska kronor
```

Kommentarer

Vi kan notera att en bättre omvandling från pdf till rå text hade varit bra. Uppdelning i block om 512 delord gjordes genom att backa till närmaste punkt före den gränsen så man fick hela meningar men rubriktexter utan punkt eller punkter mitt i meningen (t ex decimaltal eller förkortningar) gör att en lite mer sofistikerad uppdelning skulle vara lämplig.

5.2 Presentationer och Möten

Det gavs en presentation för Trafikverket rätt tidigt i projektet. Ett av syftena var att få in projektidéer. Uppslutningen blev stor och gav många kontakter. Overhead-bilderna kan ses här[13] och för varje projekt är slutsatser angivna i nedanstående avsnitt.

5.3 Ärendeklassifieringar och järnvägsväder

Utifrån väderprognoser och iakttagelser kan åtgärder behöva göras, t ex snöröjning. Likartad informationsstruktur kan även hantera andra händelser, t ex Naturhändelser i form av älgar på spåret då kanske jägare eller andra behöver anlitas. I värsta fall kan det handla om personer på spåret då polis behöver tillkallas och tåget stoppas.

Idealiskt vore ju om (lok)föraren kunde prata direkt med systemet så vi gjorde en liten test med Googles röstigenkänningstjänst (på svenska) för att få fram ord som sedan användes av BERT för att göra en skattning av vilken typ av ärende det handlar om för att snabbare hitta lämplig åtgärd. Vi gjorde några enkla prov med att läsa upp det som var antecknat men i verkligheten är nog intalade meddelanden mer informativa än de förkortade, lite kryptiska, noteringarna.

Resultatet av den lilla testen var att 87 % av gångerna hittades rätt orsakskod. Vi ser ett litet utdrag av några orsakskoder i Figur 2 samt några inrapporterade händelser i Figur 3. Som underlag fanns 2 exceldokument med ca 4 400 + 1 240 händelseexempel, vi använde en del av dem (80 %) för träning och kolumnen TRV-interntext för att skatta orsakskoder (de visas som text i högerkolumnen) på de oanvända 20 % så BERT hade alltså inte sett de händelseexemplen alls innan. Vi provade tre olika varianter av BERT; Arbetsförmedlingens, Kungliga bibliotekets samt Googles flerspråkiga BERT, de var rätt snarlika i prestanda men KB var aningen bättre. Vi passade på att använda programvaran Knime för beräkningar, plottar mm och fått viss vänlig hjälp från det svenska företaget Redfield som utvecklat BERT-modulerna som krävde

[13] https://github.com/nutte2/TRFV/blob/main/AI_NLP_TRFV_UMU_2_TS%20(3).pdf

lite specialhantering. Resultatet kan man se i Figur 4 där även felklassifieringen presenteras. En del av klasserna var sällsynta och borde hanteras via teknik för obalanserade tränings-set men det fanns inte tid för det så de klassades dåligt. Antal exempel var inte heller så högt med tanke på hur stora BERT-modellerna var så det är lätt att man överträner nätverken, dvs om man tränar lite längre så lär sig modellen att minnas exemplen och får sen svårt att generalisera till helt nya okända exempel. Det illustreras till höger i Figur 5.

Som en liten förstudie testades om man utifrån lokförarnas beskrivningar (TRV-interntext, ”Det ligger två rådjur vid stolpe 200 och ett stolpe 201.“) skulle kunna skatta orsakskoder (Nivå 3 i vårt fall, t ex ”Storm/Snöstorm“) direkt från texten.

Kommentarer

Siffran 87 % betraktades som för dålig av de som föreslog testet, för oss var den ändå imponerande bra med tanke på de rätt kryptiska texterna och rätt många händelser saknade text eller hade annan text av typen ”felet är åtgärdat“.

5.4 Dokumenthantering

Trafikverket använder massor av dokument. Dokumentklassen *Trafikverksdokument* (TDOK) består av dokument som har löpnummer, versionsnummer och en viss struktur. Med tiden har det dock blivit svåröverskådligt vilka dokument som hänvisar till vilka andra. Vi gjorde ett litet testprogram som tragglade igenom några dokument och såg vilka andra dokument de hänvisade till så man kan få en liten träd- eller snarare graf-struktur över hur de hänger ihop. De är ofta underlag för upphandlingar. En idé var att också titta igenom texten och hitta motstridiga krav, t ex kan det stå att kontaktledningen ska vara tillverkad av koppar och i ett annat dokument står det att den ska vara tillverkad av aluminium, detta kan orsaka problem om olika leverantörer tittar i olika versioner av dokument.

Exempel på utskrift från vårt lilla testsystem som hittar korsreferenser i dokument genom att leta efter TDOK årtal:nummer samt omgivande tecken så man ser ungefär vad referensen verkar handla om:

	På vagn	**JVA 03** **Järnvägsföretag, Vagn, Hjulskada**	grundhändelsen.
Slangbrott	På dragfordon	**JDM 05** **Järnvägsföretag, Dragfordon/ Motorvagn, Bromsfel**	
	På vagn	**JVA 02**	
Snö/is Fel i infrastrukturen föranlett av snö/is härleds endast till berörd ONA-kod då snöberedskapsnivån **är höjd** till 2 eller högre. Fel i infrastrukturen föranlett av snö/is då snöberedskapsnivån **ej är höjd**, orsakskodas med relevant I-kod.	Fel i infrastrukturen. **Läs informationen till vänster!**	**ONA** **Olyckor/Tillbud & Yttre faktorer, Naturhändelser** **eller I-kod**	Gäller endast vid höjd snöberedskapsnivå – annars används relevant I-kod. Var tydlig med vad som hänt, i fritexten.
	Då försening uppstår pga. snöröjning i spår eller på plattform.	**ONA** **Olyckor/Tillbud & Yttre faktorer, Naturhändelser**	Var tydlig med vad som hänt, i fritexten.
	Vid snöstorm.	**ONA 03** **Olyckor/Tillbud, Naturhändelser, Storm/snöstorm**	Var tydlig med vad som hänt, i fritexten.
Solkurva	Solkurvor orsakas nästan uteslutande av eftersatt underhåll eller avvikelser i spårets konstruktion samtidigt som stora tryckkrafter råder i spåret p.g.a. solvärme.	**IBÖ 01**	
Specialtransport	Används då specialtransporten påverkar tågföringen, exempelvis pga. annat spårval eller en punktnedsättning, och detta inte är anpassat i körplanen.	**JAS 01** **Järnvägsföretag, Avvikande sammansättning, Överskjutande lastprofil/Farligt gods**	Alltid fritext då JAS används. Kontrollera först uppgifter i **Trainplan**.

Figur 2: Exempel på orsakskoder ur Trafikverkets instruktion.

T	U	V	W	X	Y	Z
Interntext	Externtext	Infrafeltext	Infrafelrapportid	TRV-interntext	Externrubrik	
40686 skulle hämta ett trasigt fordon i S				40686 skulle hämta ett trasigt fordon i Söderhamn, men på grund av att det är för halt så får hon ju	Oväder	
			FR012	extremt halt billeberga spår 2 enligt spärrfärd/växling som kört där	Spårhalka	
			FR012	Uppställningen västerslätt, halt vid sidan av spåret.	Snö och is	
			FR012	Troligen snö eller is i spårspär 553	Snö och is	
			FR012	Växel 38 har frusit fast.	Snö och is	
			FR012	Växel 420 ej kontroll höger.	Snö och is	
Förare stannar lite extra och skrapar is				Förare stannar lite extra och skrapar is		
			FR012	Is i Tornehamnstunneln.	Snö och is	
			FR012	Is i Nuoljatunneln.	Snö och is	
				Väldigt halt på plattformen.	Snö och is	
			FR012	Spårspärr 573 strular till och från.	Snö och is	
Trafiken åter i normalläge.Inga m			FR012	Trafiken åter i normalläge.Inga mötesmöjligheter.Växel 1 ej vänster, växel 2 ej höger.	Snö och is	
Infoplan.Växlar rengjorda , möte			FR012	Kommer ej kunna hållas öppen av entreprenör, prognos måndag 15:30Inga mötesmöjligheter.Växel 1 ej höger, växel 2		
			FR012	Snö och is i växeln.Växel 2 ej vänster	Snö och is	
Flertalet växlar påverkas av snö,			FR012	Flertalet växlar påverkas av snö, innebär begränsning på trafikeringsmöjligheterna. Förseningar kar	Snö och is	
Inga mötesmöjligheter			FR012	Entreprenör på plats ca 08:40Inga mötesmöjligheterVäxel 4 ej kontroll höger	Snö och is	
Ordergivning, tågen kan tappa nå			FR012	Ordergivning, tågen kan tappa nån minut.Plk: Silvergruvvägen larmar rött.	Snö och is	
Växelfel. Inga mötesmöjligheter.			FR012	Växelfel. Inga mötesmöjligheter. Förseningar att vänta.växel 5 ej kontroll vänster och växel 1 ej hög	Snö och is	
Trafik åter i normalt läge.Inga mö			FR012	Trafik åter i normalt läge.Inga mötesmöjligheter.Växel 506 ej kontroll vänster.	Snö och is	
Trafiken åter.Växelfel. Inga mötes			FR012	Trafiken åter.Växelfel. Inga mötesmöjligheter.Växel 1 ej vänster och växel 4 ej kontroll höger.	Snö och is	
			FR012	Växlarna 2 och 1 ej rakläge.	Snö och is	
			FR012	Växel 29 ej kontroll vänster. Vill ha den klar till senast 18:50.	Snö och is	
Felet avhjälpt Felavhjälpare på pl			FR012	Felet avhjälpt Felavhjälpare på plats Tågen förväntas tappa ca 10 minuter.Växelfel. Stoppkörning.Sr	Snö och is	
			FR012	Snö och is i spårspärrenSpårspärr 553, ur kontroll i påläge.	Snö och is	
			FR012	Spårväxel 27 och 5 går ej att läggas om.	Snö och is	
			FR012	Spårspärr 32, ej på.	Snö och is	
			FR012	Växel 1 går ej att lägga om	Snö och is	
			FR012	Spårspärr 548 ej på och 573 ej ned och ej på.	Snö och is	
			FR012	Spårspärr 430 ej på.	Snö och is	
			FR012	Växel 424 ej höger.	Snö och is	
Infoplan.Växlar rengjorda, möten			FR012	Inga mötesmöjligheter.Växel 1 ej vänster. Växel 2 ej höger.	Snö och is	
8902 är första tåget att passera.			FR012	Tung, blöt snö som gjorde att bommen inte orkade gå upp. Felet avhjälpt.Felavhjälpare på plats.89	Snö och is	
			FR012	Samtliga växlar på dp behöver ses över. Troligen snörelaterat. Rings ut kl 07:00	Snö och is	
Infoplan. Växel 1 går ej i högerläg			FR012	Felet avhjälpt.Felavhjälpning påbörjadVäxel 1 går ej i högerläge	Snö och is	
			FR012	Spårspärr 438 går ej i av-läge.	Snö och is	
			FR012	Plk Kramsta larmar säkerhetsfel väg, troligtvis pga snö.	Snö och is	
			FR012	Felavhjälpare framme cirka 0740Vxl 12 ej vänster.	Snö och is	
			FR012	vxl 14 gick inte ur kontroll, rapporterar försening - funkar nu efter VXVTurbo	Snö och is	
Felet berodde på en isklump, som är bo				Felet berodde på en isklump, som är borttagen. Ingen trafikpåverkan. /TL Nord.Föraren på 41861 är	Snö och is	
Växel 32 gick ej i kontroll i vänste			FR012	Växel 32 gick ej i kontroll i vänster för tåg 40963 när de skulle gå runt med lok. Föraren fick hacka l	Snö och is	
Ingen mötesmöjlighet i Könsa pg			FR012	Växel 22 ej kontroll höger	Snö och is	
Omkodad pga blixtnedslag i isola			FR012	Omkodad pga blixtnedslag i isolator. //TRvso anm.Isolator sprucken i KattarpBytt en isolator som vi	Naturhändelse	
			FR012	Varit slyröjning på sträckan under natten, men vid km 80+200 hänger det fortfarande ut grenar son	Naturhändelse	
			FR012	Misstänkt påkörd hund vid norra växelgruppen.	Naturhändelse	
Halt på hörkenspåret.				Halt på hörkenspåret.		
			FR012	Rester av något djur som låg i diket som fåglarna åt av.kadaver?? 3 st kungsörnar äter på något äv	Naturhändelse	
Föraren sa att han tyckte det var halt i n				Föraren sa att han tyckte det var halt i några backar och att han hade ett tungt tåg. ingen bra kombination		
			FR012	Levande älg innanför stängsel vid km 25	Naturhändelse	
			FR012	Vid södra växeln ligger en presenning	Naturhändelse	
			FR012	Träd på spår. Km 812+809	Naturhändelse	
Slirade och fick backa tillbaka för att ta				Slirade och fick backa tillbaka för att ta mer fart.		

Figur 3: Några inrapporterade händelser vars TRV-interntext ska tolkas av BERT för att ge klassningen (till höger) och sen rätt kod.

Scorer View

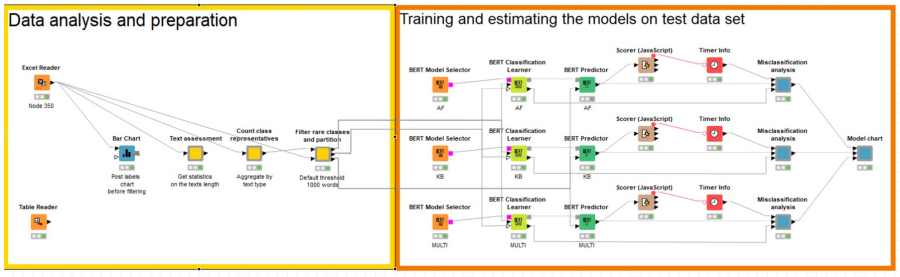
Confusion Matrix

	- (Predic…	Brand (P…	Kyla (Pr…	Skred (P…	Snö och …	Spårhalk…	Storm/S…	Översvä…	
- (Actual)	455	2	0	1	3	3	11	2	95.39%
Brand (…	5	22	0	0	0	0	0	0	81.48%
Kyla (Ac…	1	0	0	0	0	0	0	0	0.00%
Skred (A…	2	0	0	3	0	0	0	0	60.00%
Snö och…	12	0	0	1	8	0	3	0	33.33%
Spårhal…	8	0	0	0	0	1	0	0	11.11%
Storm/S…	21	0	0	0	0	0	31	0	59.62%
Översvä…	0	0	0	0	0	0	1	2	66.67%
	90.28%	91.67%	NaN%	60.00%	72.73%	25.00%	67.39%	50.00%	

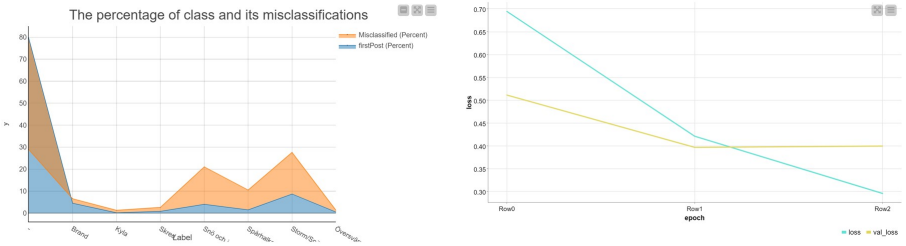
Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
87.29%	12.71%	0.601	522	76

Såhär ser beräkningsschemat ut:



Figur 4: Från en Järnvägsväderkörning där vi ser hur orsakskoder som skattats från fri text (miss)klassificeras samt en bild av hur modellen ser ut i Knime.



Figur 5: Här ser vi hur sällsynta klasser ofta klassas fel medans den vanligaste (streck = ingen info) är vanligast. Om man tränar under många epoker så minskar träningsfelet (loss) men valideringsfelet på nya okända data (val_loss) inte minskar och efter några epoker till kommer det stiga.

```
TDOK 2012:37_Metadata för digital projekthantering Väg.pdf, 17 sidor.
hanteras enligt <TDOK 2012:1171>Systemnummer och K
öljande dokument: <TDOK 2012:35>Digital projekthant
e för modell, RFM <TDOK 2015:0181>Objektorienterad i
ll, version 3.0. <TDOK 2012:36> Program och verkt

----

TDOK 2016:0407_Data och dokumentation till förvaltande system - Järnväg.pdf,128
sidor.
RVINFRA 00226 eller <TDOK 2016:0231>K8.Varje anläggnin
nar namnruta enligt <TDOK 2016:0213> Exempel på dokume
en namnruta enligt <TDOK 2019:0213> Exempel på ritnin
.TRVINFRA 00226 och <TDOK 2016:0231>gäller för teknisk
A 00226 eller A.2 i <TDOK 2016:0231>K7.Märkning och än
a namnsättas enligt <TDOK 2019:0215>3.3.3GADK1.Modell
```

Kommentarer

Det här programmet var relativt trivialt då man bara behövde söka efter TDOK årtal:nummer men även det kan vara krångligt, som så ofta vid programmering var det några bekymmer som gjorde att vi inte hann göra något ”läckert“ grafiskt: Filnamnsproblem; när man hämtade hem TDOKs, ääö blev felaktiga p g a fel teckenuppsättning för Windows, och rubrikerna gick inte att använda som filnamn; t ex är inte kolon eller / tillåtet och – var utbytt mot _. Dokumentet TDOK 2014:0162 gav felmeddelande (korrupt) när man försökte hämta det. Namngivningen var lite inkonsekvent, ibland aa ibland å och ibland gavs dokument av olika slag (krav respektive råd) samma filnerladdningsnamn, vilket orsakade namnkrockar som behövde hanteras. Sedan visade det sig att vissa TDOK-nummer var lite kortare och att dokumenten såklart hade sina egna TDOK-nummer på många ställen som behövde rensas bort.

Framtiden ser ändå ljus ut för sådana här projekt med många nya spännande möjligheter. Bl a har det nyligen (Juni 2021) kommit ut databaser på svenska för att kunna träna nätverk för att t ex se om meningar har liknande betydelse[14]. Något annat som utvecklas stark nu är att representera kunskap i grafer, Graph Neural Networks(GNNs) for Natural Language Processing[15], något som är intressant för att lagra dokumentens innehåll och kunna jämföra dem.

5.5 Tågförseningar

Här har vi använt anrop till Trafikverkets öppna Applications Programming Interface (API) för att hämta information om förseningar samt lagt in delar av den i förbestämda mallar till meddelanden som visas för allmänheten på perronger via programvaran ANNO. Trafikverkets öppna API når man genom att ansöka om medlemskap för att få ett lösenord och det är väldokumenterat. Tanken var att automatiskt generera text baserat på det som står i API:et och

[14] https://spraakbanken.gu.se/resurser/superlim
[15] https://arxiv.org/abs/2106.06090

ge förslag på text att lägga in i ANNO för senare utskick till skärmarna på plattformar i respektive stad. Det var lite trixigt att anropa från ett Python-program då mycket var baserat på programspråket Java eller andra mer vanliga dataspråk i webbsammanhang. Här är t ex koden för att hämta det fullständiga namnet på en Tågstation baserad på en intern förkortning:

```
# Making a POST request to Trafikverket API
def stationname(ls):
  # Gör om en kort unik Signatur till stationsnamn
  data = '''
  <REQUEST>
    <LOGIN authenticationkey="<borttagen av säkerhetsskäl>" />
    <QUERY objecttype="TrainStation" schemaversion="1.4">
      <FILTER>
        <EQ name="LocationSignature" value="'''
          data = data + ls
          data = data + ''''/>
      </FILTER>
      <INCLUDE>OfficialLocationName</INCLUDE>
      <INCLUDE>AdvertisedLocationName</INCLUDE>
      <INCLUDE>AdvertisedShortLocationName</INCLUDE>
    </QUERY>
  </REQUEST>
  '''
  data = str.encode(data) # Hantera Gä - encoding utf-8
  r = requests.post('https://api.trafikinfo.trafikverket.se/v1/
                     data.json', data = data)
  # success code - 200
  rj=r.json()
  return(rj["RESPONSE"]["RESULT"][0]['TrainStation'][0]
           ['AdvertisedShortLocationName'])
```

En liten film om våra korta försök kan hittas här[16].

Kommentarer

Det här fungerade ganska bra men tillämpningen är lite tveksam då Trafikverket vill separera ANNO från API-systemet då händelserna i viss mån kan ske parallellt så man vill inte behöva låta det ena vänta på det andra. Kanske kan man tänka om ifall NLP-teknik gör det mycket snabbare än en manuell operatör hinner knappa in och dessutom är det kanske inte världens roligaste arbete att upprepa inmatningen för många stationer eller delsträckor och dessutom med risk för fel. I förlängningen kanske man kan ersätta det begränsat antal meddelanden mallarna kan erbjuda och få lite mer fri och förståelig text. En vän råkade t ex ut för meddelandet: ”Tåget mellan Umeå och Stockholm är inställt av planeringstekniska skäl“ en timme innan avresa och det var spekulationer om vad det kunde tänkas betyda. Kanske nåt problem med lokförarbyte? och utrycket kunde vara användbart privat; ”planeringstekniska skäl” ska jag använda nästa gång frun undrar varför diskmaskinen inte är tömd.

[16] https://www.youtube.com/watch?v=YTbJIq4lbLI

5.6 Vattenanmälningar

I den här delen av projektet har vi hämtat information från dokument för att skapa underlag för att fylla i Länsstyrelsens webbformulär angående vattendrag och broar man önskar bygga m m. Den information som behöver hämtas in kan handla om beräknade flöden och en mängd annan data inkl bilder och kartor. Tanken är att man jobbar fram en förhållandevis generell beskrivning av den typ av information man önskar hämta från dokument för att sen ”köra“ beskrivningen och på det sättet automatiskt få in informationen i formuläret. Det handlar om ungefär 20 informationsbitar för att fylla i Länsstyrelsens sjusidiga online formulär. Helst kulle man vilja göra det automatiskt via t ex Web-traversal-library (WTL), men det blir antagligen för många fel om man gör på det sättet, så vi jobbar på att ta fram ett skräddarsytt system som levererar ett tidsbesparande underlag som sedan manuellt kan granskas, justeras och kompletteras.

Rubrik	Hittas	Kommentar
Länsstyrelse	Rubrik MKB	
Kommun	Rubrik MKB	
Typ av sökande	Myndighet	Samma varje gång
Organisationsnummer	xxxx	Samma varje gång
Typ av verksamhet	Valt byte av trumma	testexempel
Medelvattenföring MQ	PM Avvattning	
Trummans diameter	PM Avvattning	
Trummans längd	PM Avvattning	
Typ av trumma	PM Avvattning	
Typ av vatten	Här bestämmer vi	Bestämmer vi som standard
Beskrivning	MKB alt PM Avvattning	
När	Manuellt	
Beskriv omgivning	MKB	
Påverkan	Manuellt	
Skyddsåtgärder	MKB	
Föroreningar	PM Markmiljö	Svår, tror det är svårt att koppla
Sedimentprover tagna?	PM Markmiljö	Kartreferens- går det söka?
Inom naturskyddat område	MKB/PM Naturvärdesinventering	Kan vi hitta en sån specifik uppgift?
Vilket vattendrag	MKB	
Koordinater	Manuellt	
Översiktskarta	MKB	
Detaljkarta	Manuellt	Även fastighetskarta och fastighetsägare
Ritning	Manuellt	

Figur 6: Här ser vi några exempel på behövlig information samt varifrån den kan hämtas

Exempel på hur frågespråket kan se ut i Python-stil (tidig version) finns i nedanstående text:

```
def vattenanmälan():
    # Hämta lämpliga dokument
    url= 'https://www.trafikverket.se/'# sajt
    # Man kan nog (annars) söka på ordet Miljökonsekvensbeskrivning
    # för att hitta rätt dokument
    mkb=frånWebsida(url,'miljokonsekvensbeskrivning5.pdf')
    PMAvvattning=frånWebsida()
    PMMarkmiljö=frånWebsida()
    PMNaturvärdesinventering=frånWebsida()
    planbeskrivning=frånWebsida(url,'180704_planbeskrivning.pdf')
    översiktskarta=frånWebsida(url,'180704_oversiktskarta.pdf')

    # Se vilka bäckar som finns nämnda och kan vara aktuella, gör en ansökan
    # för varje.
    # ner=named entity recognition, letar efter platser med ordet bäck i sig,
    # man kan ha flera varianter
    bäckar=ner(mkb,'POS','*bäck*||*ån*||*älv')
    for bäck in bäckar:
        ansökan=formulär('Länsstyrelse vattenanmälan v1',bäck)
        # Skapa grundformulär

        # Fyll formuläret med massa fakta via add (addera)
        ansökan.add('Länsstyrelse',ner(mkb,'ORG','*länsstyrelse*'))
        ansökan.add('Kommun',ner(mkb,'POS','*kommun'))
        # hitta mer eller mindre googlar fram resultat
        ansökan.add('Organisationsnummer',hitta(['organisationsnummer',
            ansökan.data('Länsstyrelse')))
        ansökan.add('Typ av verksamhet','Byte av trumma')

        #hittafakta använder frågor&svar för att hitta info
        ansökan.add('Medelvattenföring MQ',hittafakta(PMAvvattning,
            ['Medelvattenföring' bäck])
        ansökan.add('Trummans diameter',hittafakta(PMAvvattning,
            ['Trummans diameter' bäck])
        ansökan.add('Typ av vatten', 'Ytvatten')

        # sammanfatta använder BERT för att hitta viktiga meningar i ett stycke
        # där sökorden eller ord med liknande betydelse finns
        ansökan.add('Beskrivning', sammanfatta(mkb,[bäck 'beskrivning'])
        ansökan.add('Beskriv omgivning', sammanfatta(mkb, [bäck 'beskrivning'
                                                           'omgivning'])

        ansökan.add('Föroreningar',hittafakta(PMMarkmiljö,'Föroreningar'))
        ansökan.add('Vilket vattendrag',bäck)
        ansökan.add('Koordinater',bäck) # behöver ofta matas in manuellt

        # figur letar fram en bild i ett dokument med egenskaper som känns igen
        # av ett AI-system, google translate (typ-ordet karta till map) +
        # deep learning (image_net) här
        ansökan.add('Översiktskarta',figur(mkb,['karta','legend'],sida=1:5))
        ansökan.add('Detaljkarta',figur(mkb,['karta','legend'],sida=6:))
        ansökan.add('Ritning',figur(mkb,['ritning','legend']))

        # redigera presenterar det man kommit fram till för ändringar och
        # kompletteringar kanske är via Anteckningar eller möjligen Word.
        # Bildfiler ligger separat för inspektion/val/ändring/uppladdning
```

```
        if ansökan.redigera()): # Om redigeringen är klar/ok så skickas den in
            if ansökan.fylli(): # Fyller i Länsstyrelsens formulär
                if ansökan.ok(): # om det verkar rätt ifyllt
                    if ansökan.betala('betaluppgifter'): # så betala
                        ansökan.klar()
```

5.7 Generering av text via GPT-2

GPT-2 är inte särskilt bra, speciellt inte på svenska men vi gjorde några försök att generera text relaterat till en planbeskrivning via viss träning. GPT-3 hade varit språkligt bättre men även där är det mer av en hitte-på text utan riktiga fakta:

```
Detaljplan för del av fastigheten Anumark 1:20, m.fl. (Dåvamyran), antagen
2005-02-16.
Byggande av järnväg får inte ske i strid med gällande detaljplan. Umeå kommun
kommer därför att ändra eller upphäva de delar av detaljplanerna som berörs av
järnvägen.
10.2 Genomförande
10.2.1 Organisation
Trafikverket ansvarar för upprättande och granskning av järnvägsplanen.
Genom järnvägsplanens samrådsprocess får Länsstyrelsen, utveckling
av samrådsförhållanden, utveckling av carnen uppstår samt carine som
klimatpåverkan.
Trafikverket får samt pandavfaller, kommunen även förligrav på falsämna både
och köksmänner landskapcksmassor att de är särskilt skyddsvärda deponi på
om planerad järnväg. De deponi på om plan ersättningsväg ägs av köksmänner.
Avvägningar på deponin på grundvatten och har uppfyllelse av projektspecifika
mål.
```

5.8 Andra idéer

Vi har även tittat på lite andra idéer, t ex hur man kan hämta ut/extrahera bilder med önskat innehåll (t ex karta) ur pdf-filer, hur man kan generera bilder baserade på en text, hur kan man snabba upp Python-program så man kan minska behovet av GPU-kort, studerat kommersiella Chat-bottar, hur man kan sortera mail utifrån textinnehållet och många andra saker.

6 Diskussion

6.1 Prototypsystem

Vi håller på att göra ett prototypsystem med en interaktiv server som illustrerar några av de försök vi gjort och tanken är att man ska kunna provköra själv och med korta svarstider. Den är åtkomlig dagtid under begränsad tid (troligen eftermiddagar i November 2021) på http://prorok.ownit.nu/ eller ev ändrad adress given via https://github.com/nutte2/TRFV, hoppas du tar chansen att prova!

6.2 Chat-bottar?

En naturlig fortsättning är att implementera Chat-robotar, system där man kan konversera med datorn för att även kunna ställa följdfrågor. Då behöver tidigare angiven fakta kunna lagras under en längre konversation. Det finns en del nya försök men många av de kommersiella systemen bygger på att någon matar in en stor mängd frågor och svar för att det ska fungera bra och det är tid och resurskrävande och nog endast realistiskt för mycket stora företag.

6.3 Strömförbrukning

System med många GPU-enheter kräver mycket effekt; ca 300 W/styck och det kan vara tusentals som ska köras veckovis vid träning. När DeepMind spelade brädspelet GO kostade varje match 30.000 i elräkning.

6.4 Molntjänster

Flera företag tillhandahåller AI-system för uthyrning på tidsbasis, t ex Azure från Microsoft. Det kan vara kostnadseffektivt men såvitt Kalle vet så har de ännu inte avancerade möjligheter för svenska språket och när/isåfall kommer nog svarstiderna bli många sekunder eftersom stora språkmodeller behöver kopieras och startas. Andra problem kan vara personuppgifter som t ex GDPR eller andra lagar såsom Act of digital Services, Lagen om digitala tjänster[17] och andra lagar, t ex för Amerikanska företag som i viss mån gäller även i Sverige.

6.5 Anonymisering av dokument inför utlämning

Offentlighetsprincipen och andra lagar kan kräva att man lämnar ut fakta till allmänheten och journalister och för att följa dessa lagar kan en anonymisering behövas och då används våra tekniker (NLP) i vissa fall av t ex sjukhus.

7 Framtidens möjligheter

Det är inte lätt att sia om vad som händer härnäst i detta mycket expansiva forskningsfält, många av de metoder som fungerade bra våren 2021 är omoderna eller inte ens längre aktuella hösten 2021. Ett sätt vi noterat det i vårt projekt är att programkod som fungerade bra i April 2021 inte längre fungerar i September 2021 då så mycket hunnit ändrats, därför lägger vi ut källkoden separat[18] så vi kan hålla den uppdaterad i alla fall under en tid.

[17] https://ec.europa.eu/info/strategy/priorities-2019-2024/europe-fit-digital-age/digital-services-act-ensuring-safe-and-accountable-online-environment_sv

[18] https://github.com/nutte2/TRFV

Del III
Begrepp och metoder

8 Begrepp

Begreppen som beskrivs nedan är mest från traditionella metoder. 2017 lanserades den redan nämnda transformern som revolutionaliserade hanteringen av naturlig text främst genom att på ett flexiblare sätt än tidigare se till hela meningar eller stycken istället för enstaka ord. Dock fortsätter begreppen nedan att vara viktiga eftersom de används i olika sammanhang och syften.

8.1 N-Gram

Ett N-gram är en sekvens av N ord i den ordningen de förekommer i meningen. N-grams används för att förstå i vilken ordning ord kan dyka upp i en mening. Meningen *Kalle reser till Umeå* innehåller tre 2-gram (bi-gram): Kalle reser, reser till, till Umeå. Tri-grams är när man grupperar ihop tre ord: Kalle reser till, reser till Umeå. Statistik över vilka N-gram som brukar förekomma i korrekta meningar är mycket värdefulla. T ex borde bi-grammet *spelet flyr* vara extremt mycket ovanligare än *viltet flyr*, information som kan användas för att undvika misstag som det som nämnts i fotnot 4.

8.2 Stemming

Genom att ta bort delar av ord, t ex pluralform eller olika verbform kan man hitta det grundord (stam) som ordet stammar ifrån för att på så sätt kunna hantera flera varianter på ett likarartat sätt. Man bör dock också vara medveten om att man samtidigt tappar information. Exempel är universitet, universitetens som kanske kan anges som universit (det behöver inte vara ett korrekt ord). Stemming används inom många sammanhang, bl a i webbsökningar där sökningar efter *bil* och *bilen* ger i stort sätt samma resultat.

8.3 Lemmatisering (lemmatisation)

Lemmatisering är besläktad med stemming men svårare. Målet är att föra tillbaka ett ord till den form man t ex skulle hitta i ordboken. Detta är ordets lemma. För att hitta det behöver inte bara själva ordet utan också kontexten tas hänsyn till. Exempelvis är lemmat till *bilar* i meningarna *Vi bilar till Stockholm* och *Vi stod i en kö med många bilar* olika. I den första meningen är det verbet *bila* och i den andra substantivet *bil*.

8.4 Påse med ord, Bag-of-words (BOW)

I vissa sammanhang, t ex när man vill bedöma om en mening uttrycker en positiv eller negativ åsikt, kan det ibland räcka att man samlar orden i ”en påse“, dvs

man bryr sig inte om ordföljden utan ser bara på vilka ord som förekommer i meningen. På så vis förenklas problemet även om potentiellt viktig information går förlorad.

8.5 RNN

RNN står för Recurrent Neural Networks; det mest kända exemplet är LSTM (Long-Short Term Memory). Det handlar sig om en typ av ANN som återkopplar till sig själv, en teknik som tidigare användes flitigare, men på grund av att dessa nätverk har svårt att hantera längre meningar utan att tappa precision har dem i de flesta sammanhangen ersatts av transformern.

8.6 Stoppord

Vissa ord, ofta korta såsom ”och“, ”att“ osv, är inte särskilt betydelsebärande och sorteras därför ofta bort i traditionell språkteknologi innan den huvudsakliga bearbetningen sker. Dessa ord kallas för stoppord. Utöver det är det vanligt att alla ord görs om till gemener för att minska på ordförrådet. Nackdelen är att man tappar de ledtrådar stora versaler kan ge. Exempel som illustrerar det är ordparen Hans/hans, Per/per och Lina/lina som felaktigt skulle betraktas som samma ord.

8.7 TF-IDF

För att klassificera dokument är det viktigt att identifiera de ord som är mest kännetecknande för ett givet dokument. För att få ett mått på varje ords viktighet i en text, ett vanligt sätt är *Term Frequency–Inverse Document Frequency*. Tanken är att ett ord är mer karakteristiskt för ett dokument ju oftare det förekommer i dokumentet (*term frequency*) och ju mer sällsynt det är i den betraktade typen av dokument (*inverse document frequency*). T ex brukar ordet ”jag“ inte vara särskilt sällsynt och därmed inte så kännetecknande i romaner även om det förekommer 1000 gånger i en given roman. Däremot är det väldigt sällsynt i vetenskapliga artiklar, vilket gör att även en enda förekomst av ordet ”jag“ i en vetenskapliga artikel låter artikeln sticka ut.

8.8 Inbäddning (embedding)

Då ANN enbart kan hantera tal avbildas ord till (vektorer av) reella tal. Ordet *Trafikverket* kan då t ex representeras av $(0.5, 22.3, 0.001)$. Denna typ av representation av ord genom tal kallas *inbäddning*. Dessa inbäddningar lär sig datorn ofta automatiskt och så att talen reflekterar vissa egenskaper av det motsvarade ordet eller samband mellan ord. Ofta har man en begränsad mängd såsom t ex 10 000 st i sitt ordförråd, som kompletteras med några specialnummer som indikerar Start ⟨STRT⟩, Stopp ⟨STOP⟩ och okänt, unknown, ord ⟨UNK⟩ eller out-of vocabulary ⟨OOV⟩.

8.9 Word2Vec

Word2Vec är ett av de tidigare inbäddningarna; varje ord omvandlas då till en mångdimensionell vektor av reella tal (typiskt 300 dimensioner). Orden hamnar då i en rymd där man kan bedöma hur nära de är varandra (dvs hur snarlika de är i betydelsen) och man kan göra vektorberäkningar av typen vad blir Kung − Man + Kvinna? Jo, det blir Drottning. Andra exempel är Paris − Frankrike + Sverige som ger Stockholm och Sushi − Japan + Sverige ger (lite oväntat?) Pizza eller Tacos.

8.10 Fasttext

Tomas Mikolov vidareutvecklade Word2Vec till Fasttext[19] som har fördelen att det snabbt (minuter) kan tränas upp på vanliga datorer och det finns färdigtränande modeller på en mängd (ca 157) språk som kan anpassas (specialtränas) för olika syften. Man använder delord vilket underlättar då t ex Universitet, universitetens och universitets hanteras tillsammans. Kalle har tidigare använt detta för att sortera till kommunen inkommande epost för vidarebefordran till ämnesmässigt lämplig handläggare efter att ha tränat nätverket på 1,2 miljoner epostmeddelanden.

8.11 Transformers, t ex BERT, XLM-R (RoBERTa)

Fördelen med transformerbaserade arkitekturer är att man studerar ord i deras kontext, vilket gör att inbäddningen t ex kan skilja mellan den *banan* som är frukten (egentligen bäret) och den man kör på. En annan viktig sak är att man använder delar av ord, vilket medför att det normalt inte finns några luckor i ordförrådet, alltså okända ord som hade saknats i en inbäddning av äldre typ som Word2Vec.

Ett vanligt sätt att träna upp dessa nätverk är att man samlar stora mängder med text, t ex hela Wikipedia, och tränar nätverken genom att dölja vissa av orden som nätverket tränas att gissa rätt.

Det finns varianter som klarar många språk (Multilingual-BERT) och specialtränade för vissa språk som t ex KB/bert-base-swedish-cased som ofta är något bättre.

8.12 Fler Transformers, t ex GPT-2, GPT-3, T-NLG, T5, mT5

Det utvecklas flera extremt stora nätverk för att t ex automatisk kunna generera tidningsartiklar eller föra intressanta resonemang. Det behövs enorm datorkapacitet till höga kostnader så en del av leverantörerna har spärrat tillgången för allmänheten och har köer och betalabonnemang. En sak som blivit lite större nu (augusti 2021) är möjligheten att automatgenerera delar av datorprogram.

[19] https://fasttext.cc/

Den populära GPT-3[20] kommer antagligen att efterträdas av GPT-4. Några av de förväntade skillnaderna mellan GPT-4 och GPT-3 beskrivs i fotnoten[21]. GPT-3 har kommersialiserats men det finns numera en snarlik open-source lösning GPT-Neo[22], och GPT-J som kan användas om man har rätt typ av hårdvara. Utvecklingen av Microsofts T-NLG verkar ha stannat av. T5 och den multispråkiga mT5[23] känns lovande, den klarar ca 100 språk inkl svenska och tillhandahålls i flera storlekar (Small (300M parametrar), Base (580M), Large (1.2B), XL (3.7B), och XXL (13B)) så man kan välja prestanda utifrån den minnesstorlek man har tillgång till, obs att parametrarna är flyttal som tar 8 byte/styck, vi har använt Base-varianten på grafikkort med 32 GB minne (Nvidia V100) och omträningen tog några enstaka timmar på de enkla försök vi gjort. Dessa nätverk är ofta av sekvenstyp, dvs man matar in en sekvens av t ex bokstäver och ut kommer en (annan) sekvens. För att ange vilken uppgift man önskar utföra så föregås insekvensen av ett prefix man hittar på själv vid träning och sedan använder (följt av ett kolontecken) på det omtränade nätet (via transfer learning), t ex ”översätt till tyska: Jag är glad“ och förhoppningsvis kommer det tyska uttrycket ut från modellen efter att man tränat modellen med ett antal meningspar. Eftersom nätens förkunskaper är så goda brukar det inte behövas så många exempel, man pratar om Zero-shot generation.

8.13 Tokenizer

Löpande text behöver delas upp i mindre bitar, eng. tokens; ibland är det uppdelat i ord men det kan vara andra sätt t ex i delord som i BERT-fallet. Det gör att man blir mindre beroende av stemming och lemmatisering då det sköts ”automatiskt“ i och med att man redan delar upp orden i kortare generellare delar.

8.14 Reguljära uttryck

Reguljära uttryck är en metod för att beskriva systematiskt uppbyggda texter, t ex datum, klockslag och GPS-koordinater. För den som inte är van vid reguljära uttryck kan de vara ganska kryptiska. Vi har använt sådana för att känna igen TDOKs-hänvisningar i text.

8.15 Huggingface

Ett mycket värdefullt ställe att hålla koll på är sajten Huggingface[24]. Där lägger många upp sina språkmodeller (t ex hittas Kungliga Bibliotekets BERT-varianter där) för mer eller mindre fri nedladdning och användning på ett nå-

[20] https://towardsdatascience.com/gpt-3-a-complete-overview-190232eb25fd
[21] https://towardsdatascience.com/4-things-gpt-4-will-improve-from-gpt-3-2b1e7a6da49f
[22] https://www.eleuther.ai/
[23] https://arxiv.org/abs/2010.11934
[24] https://huggingface.co/

gorlunda standardiserat sätt. Huggingface är nu med i BigScience[25], en stor europeisk satsning på gemensamma lösningar och väldigt kraftfulla datorer för språkmodeller på andra europeiska språk än engelska.

8.16 Semantic Textual Similarity (STS)

Med Semantic Textual Similarity (STS) menas hur lika ord, meningar eller texter är med avseende på innebörd även om de kanske ser väldigt olika ut. Eftersom syftet med en språkmodell är att i så stor utsträckning som möjligt ”förstå“ språk kan en bra språkmodell användas som bas för att lära sig inbäddningar som representerar innebörden av en mening. Ett flertal svenska modeller som ska klara STS är på gång. Ett exempel är BERT-Base-Swe-CT-STSb, som baseras på tekniker publicerade i [CGG+21]. Det är intressant att unga svenska forskare bidrar till arbete och innovationer som normalt görs av stora företag som Google eller Facebook. Källkoden och modellerna kan nås[26] och sen testas på egna projekt. Företaget Peltarion har infört möjlighet att använda en hel del AI och NLP inkl detta i sin utmärkta platform[27].

8.17 Hård- och mjukvara

För att snabba upp beräkningarna använder man normalt ett beräkningsgrafikkort från leverantören Nvidia som innehåller ett stort antal (flera tusen) grafikprocessorer (GPUer). På så sätt kan man använda konsumentkort för AI-beräkningar som är förhållandevis prisvärda eftersom de tillverkas i stora serier för datorspelbranschen. Ska korten användas i en webbserver gäller speciella regler (EULA) och man behöver normalt köpa specialversioner för det ändamålet. Tillhörande mjukvara för träningen av nätverken på GPUer är vanligen Tensorflow eller PyTorch; för just texthantering tycks PyTorch nu ha övertaget. Det vanligaste programmeringsspråket är det tolkade (interpreterande) skriptspråket Python men rutiner skrivna i andra språk, t ex i C/C++ kan anropas och exekverar vanligtvis mycket snabbare då de är kompilerade. Python har ett stort antal tidsbesparande färdigskrivna bibliotek som förenklar programmerandet, vilket är en av huvudorsakerna till språkets popularitet plus att det är ganska lätt att lära sig och effektivt att skriva kod i.

9 Programspråk, -bibliotek och -system

För att förstå detta kapitel är det bra om man känner till lite mer om programmering. Vi har använt dataspråket Python som är det vanligaste och kraftfullaste inom detta område eftersom det har så många färdiga användbara bibliotek som då gör det snabbt att konstruera program i. Däremot exekverar (körs) programkoden lite långsammare än de flesta andra språk så för produktion kan man

[25] https://bigscience.huggingface.co/en/#!index.md
[26] https://github.com/FreddeFrallan/Contrastive-Tension
[27] https://peltarion.com/integrations

anpassa koden på lite olika sätt, Pypy, Cython, parallelisering och en mängd andra möjligheter.

9.1 Python-programbibliotek som är bra att känna till

Vi har använt alla dessa under projektets gång med mer eller mindre bra framgång. Det finns förstås många flera att pröva/utvärdera för den intresserade.

9.1.1 Tensorflow/Keras

Tensorflow (TF) utvecklas av Google och är i skrivande stund uppe i version 2.4. För att förenkla användningen av TF har Keras tagits fram som gör det ganska lätt för nybörjare att skapa AI-modeller inom t ex bildigenkänning mm.

9.1.2 PyTorch

PyTorch utvecklades först av Facebook och nu har Microsoft kommit in som medutvecklare. PyTorch har en lite högre inlärningströskel då man måste göra en del saker själv (differentiering t ex) men det är å andra sidan kanske lättare/flexiblare att använda (än TF) då/om man vill prova egna, kanske nya, typer av Artificiella Neurala nätverk (ANN).

Vill man köra PyTorch på Googles utmärkta och snabba TPU-hårdvara (som normalt används för Tensorflow) så går det med viss möda på t ex Colab[28] och med t ex Python-biblioteket XLA[29] Pytorch kom i en ny version 1.9 21 juni 2021 på vägen mot 2.0.

9.1.3 Spacy

Det finns ett användbart gratis grundbibliotek för NLP-hantering, Spacy, och i det kan man dela upp text i ord, ordklasser (Part-of-speeech, POS), Named-Entity-REcognition (NER), likhet, text klassificering och många andra saker.[30] [31]

När man hanterar olika bibliotek och flöden mellan under olika inlärningsförlopp och kodblock används ofta tekniken pipelines för att knyta ihop dem:

```
from transformers import pipeline
```

9.1.4 NLTK

Ett vanligt standardbibliotek för textbehandling tidigare är gratisbiblioteket Natural Language Toolkit (NLTK)[32] men det baseras på en bok från 2009 och

[28] https://medium.com/pytorch/get-started-with-pytorch-cloud-tpus-and-colab-a24757b8f7fc

[29] https://github.com/pytorch/xla

[30] https://www.kb.se/samverkan-och-utveckling/nytt-fran-kb/nyheter-samverkan-och-utveckling/2020-11-23-spacy---ny-svensk-modell-for-storskalig-textanalys.html

[31] https://spacy.io/api/kb

[32] https://www.nltk.org/

tycks inte ha utvecklats nämnvärt sedan 2019 så inte nåt med transformers men det finns en del användbara orddatabanker (corpora).

9.1.5 fitz

Det finns många Python-bibliotek som konverterar pdf-filer till rå text för analys. Bland det trixiga finns att på sidorna kan finnas texter längst upp eller längst ner, sidnummer, tabeller etc som behöver hanteras så inte språkmodellen tappar bort sig. Det gäller också att hålla reda på teckenuppsättning för t ex pdf eller text från webben så att våra åäö blir rätt.

9.1.6 docx2python

Det går även bra att läsa in Microsoft Word/docx-filer till rå text för vidare hantering i Python.

9.1.7 Tika

Läser pdf och en mängd andra format via en Java-server som man kan anropa. Konverteringen går rätt fort när servern väl är uppstartad och resultatet ser bra ut, dock saknas möjligheter att styra hur den går till så man får med rubriker och andra saker på det sätt man önskar.

9.1.8 pandas

Ett välkänt och kraftfullt bibliotek för att enkelt/praktiskt importera och hantera främst numeriska data för Python-program.

9.1.9 urllib

Innehåller moduler för att hantera URL:er[33]

9.1.10 Web Traversal Library (WTL)

Ett användbart bibliotek för att interagera med webbsidor är WTL; Utvecklat av en Klarna-inkubator[34] och baseras på det mycket användbara och kraftfulla webbautomatiseringsbiblioteket Selenium[35].

9.1.11 bs4

Beautiful soup kan låta smaskigt men är (ännu) ett bra bibliotek för hantering av webbanrop direkt från Python.[36]

```
import BeautifulSoup
```

[33] https://docs.python.org/3/library/urllib.html#module-urllib
[34] https://github.com/klarna-incubator/webtraversallibrary
[35] https://en.wikipedia.org/wiki/Selenium_(software)
[36] https://www.crummy.com/software/BeautifulSoup/

9.1.12 PySide2

För att skapa användargränsnitt (Graphical user interfaces, GUI) för program som ska användas lokalt på datorn finns några olika bibliotek. TkInter är ett i Python inbyggt men med rätt torftigt utseende/hanteringsbibliotek så många rekommenderar det utmärkta PyQt5/6 som dock är kommersiellt. Ett nästan identiskt men fritt är Pyside2. import QtGui - (Pyside2 är en gratisversion av PyQt5)

9.2 Webbtjänster

För att göra program åtkomliga för andra kan man antingen sälja/ge bort program för lokal installation vilket kräver vissa insatser av användarna samt ställer höga krav på användarnas hårdvara i vårt fall. Ett annat alternativ är att tillhandahålla molntjänster där användarna kan köra på, ofta stora företags, servrar via webb-anrop. Några exempel är Azure och Colab. Några har sekundpriser och andra är gratis, kanske för studenter men även för allmänheten. I vårt fall är start-up-tiden för att ladda upp/starta språkmodeller rätt hög, det kan ta nån minut och det är ofta för lång tid för att användare ska ha tålamod att vänta. De generella molntjänsterna kan ju inte ha språkmodellerna förinladdade och startklara så vår idé är att dedicera en dator som har språkmodellerna förladdade och är beredd på att svara snabbt, förhoppningsvis inom tio sekunder så det blir användbart i praktiken.

Det finns olika sätt att göra detta på men vi har valt att som hårdvara testa en Linux-server med ett avancerat Nvidia-kort som är tillåtet att använda för serverbruk enligt deras licenskrav. Som mjukvara är Python självskrivet men sen finns det lite varianter för hur man ska göra funktionaliteten åtkomlig, grundplanen är att implementera FastAPI då det är känt för att vara modernt och snabbt men kan behöva ändras. Några andra problem som kan dyka upp är certifiering av https, dynamisk IP för servern och hur man ska hantera flera anrop samtidigt då det bara finns ett beräkningsgrafikkort.

9.2.1 REST (REpresentational State Transfer)

Anropstyper till webb-servrar kan vara lite olika, vi har valt det omtyckta REST.

9.2.2 Django

Klassiskt har många webb-tjänster använt Django som innehåller mycket av det man kan behöva, t ex en databas. Många stora och små företag använder det men anses av en del att vara klumpigt och gammeldags.

9.2.3 Flask

Flask är en populär lättviktsmiljö, dvs rätt enkel start men man behöver lägga till funktionalitet själv allteftersom behovet uppstår. Ett förslag på hur man kan

göra ett ML API finns här[37]

9.2.4 FastAPI

Ett modernt, snabbt och stigande i populatitet alternativ är FastAPI. Det verkar också vara ovanligt enkelt att använda. En av finesserna är att man kan få automatiskt kontroll av parametrar och datatyper via type-hinting i nyare versioner av Python (3.6+).[38] [39] [40] [41]

9.2.5 Docker

Man kan använda Docker, ett slags container, för att distribuera FastAPI-funktionalitet[42] men det går även med andra varianter, t ex Deta[43] [44].

Exempel på hur man kan köra FastAPI på en egen Ubuntu-server med programmet uvicorn kan man läsa om här[45]. Vill man lägga till HTTPS-funktionalitet[46] kan man t ex använda Let's Encrypt.

9.2.6 HTTPS

För att använda säkerhets-surfningen HTTPS behöver man få ett certifikat utfärdat, t ex från Let's Encrypt[47] som tillhandahåller det gratis.

9.2.7 requests

För att göra anrop till webb-sidor för att t ex hämta filer eller liknande kan Python-biblioteket requests vara användbart.

När det gäller HTML-protokollet används GET, PUT, POST och DELETE.

9.3 Lite andra nyttigheter

9.3.1 Användargränssnitt

För att skapa dialog-rutor i rena Pythonprogram finns ett antal programbibliotek.

För webbgränssnitt finns t ex PyWebIO[48]

[37] https://towardsdatascience.com/how-to-build-a-machine-learning-api-using-flask-2fb345518801

[38] https://betterprogramming.pub/why-you-should-stop-using-flask-and-start-using-fastapi-1a091bfc2fbf

[39] https://fastapi.tiangolo.com/tutorial/first-steps/

[40] https://tiangolo.medium.com/introducing-fastapi-fdc1206d453f

[41] https://github.com/tiangolo/fastapi

[42] https://fastapi.tiangolo.com/deployment/docker/

[43] https://fastapi.tiangolo.com/deployment/deta/

[44] https://blog.deta.sh/posts/deta_fastapi_jwt_auth_part_2/

[45] https://stackoverflow.com/questions/62898917/running-fastapi-app-using-uvicorn-on-ubuntu-server

[46] https://fastapi.tiangolo.com/deployment/https/

[47] https://letsencrypt.org/getting-started/

[48] Abdishakur. Create Interactive Web Applications in Pure Python with PyWebIO, 2021

9.3.2 Knime

En miljö för att hantera Data Science problem som Machine Learning (ML) och liknande är Knime[49]. Det består av moduler, beräkningsklossar, med in- och utgångar som kan kopplas ihop till ett överskådligt schema. Nyligen har det kommit moduler även för NLP och BERT som vi testanvänt i ett av prototypprojekten. Det är trevligt och bra samt gratis om man kör det på en egen/enstaka maskin men kostar om det ska ligga på en server. Ett problem om man forskar är att det kanske inte finns några moduler med den senaste tekniken och då behöver man anlita någon eller konstruera dem själv, t ex går det bra att formulera dem i Python.

9.3.3 Dotsity

Programvara för att generera diagram med cirklar och pilar, t ex för att ange förkunskapskrav och kurser. Vi tänkte att man ev kan använda det för att se vilka dokument som refererar till varandra på ett överskådligt sätt.[50]

9.3.4 Word clouds

För att visualisera de vanligaste orden så kan man skapa en figur där storleken på bokstäverna är i proportion till vanligheten. Hur detta kan göras i Python står här[51].

9.3.5 Talat språk, speech recognition

Lite utanför vårt projekt men inte omöjligt. Det finns Python bibliotek för det (men inte på svenska än):

```
as sr
r = sr.Recognizer()
with sr.Microphone() as source:
```

Igenkänning av pratad svenska kan man göra men inte gratis just nu, kanske att Microsoft får till det i sitt officepaket senare. Det finns en google-tjänst; 1 timme/månad gratis, sen ca 10 öre per påbörjad 15 sekunder (ca 50 öre/minut).

Det görs nu försök av bl a KB med igenkänning av talat svenskt språk till text (ASR) så vi får se när det blir klart till användning.[52]

9.3.6 Bilder till språk

CLIP är model från OpenAI som matchar bilder med text[53] och det finns flera liknande.

[49] https://www.knime.com/
[50] https://dotsity.com/
[51] https://towardsdatascience.com/simple-wordcloud-in-python-2ae54a9f58e5
[52] https://huggingface.co/KBLab/wav2vec2-large-xlsr-53-swedish?fbclid=IwAR1DpuSwkl1hTeJMd6K6ZJzmzzgFvl2ic6h5H0V-p8biM5phhqD2s7j-_0k
[53] https://github.com/FreddeFrallan/Multilingual-CLIP

9.3.7 Utvecklings och leveransmiljöer

Många av biblioteken inom NLP behöver Linux för att fungera, de flesta fungerar även under Mac OS X men Windows 10 är tyvärr inte alltid användbart, t ex saknas då stöd för Tiki och requests.

Exempel på speciellt bra Linux-kommandon finns här[54].

Det finns några olika sätt att köra Linux tillsammans med eller under Windows; man kan köra dem genom att välja vilket som önskas vid uppstart av datorn, så kallad Dual boot, vi har provat några varianter av det men det är ganska otillförlitligt; en uppdatering i det ena (Windows) eller andra (Linux) kan göra dual-boot-inställningen oanvändbar eller rentav göra hela maskinen oanvändbar och i behov av total nyinstallation.[55]

En annan variant är att installera ett subsystem, app (Windows subsystem för Linux, WSL 2) i Windows så man startar Windows som vanligt och startar sedan ett kommandofönster med Linux i önskad variant (t ex Ubuntu). Det finns mer att läsa om WSL 2[56] och det utvecklas aktivt.

Tidigare och även nu kan det vara svårt att nå beräkningsgrafikkort/GPU via CUDA från WSL 2 men nyligen (mars 2021) verkar det fungera med ev buggar, läs om CUDA i WSL 2[57] samt i[58].

Docker kunde tidigare inte köra GPU under Windows/WSL 2 men det verkar vara löst nu (maj 2021). Läs om GPU under WSL 2 via Docker[59].

För att undersöka/testa webbmiljöerna kan man använda POSTMAN och/eller curl[60].

10 Resultat

Ett av de största resultaten av vårt projekt var att vara väl insatt i tekniken och sprida kunskapen så den kan användas på skarpare projekt, bland annat via seminarier, publika kodexempel och denna rapport med många länktips. Vi har gjort några generella tester och experiment, genomfört några prototypprojekt utifrån inkomna idéer och, lite försenat pga sen hårdvara, tänker vi oss att slutföra ett testprojekt med Vattenanmälningar samt få till en AI-språk-server samt en samling beskrivna kodexempel trots att projekttiden egentligen är slut nu i augusti 2021. Det har varit intressanta och inspirerande projektmöten varannan vecka.

[54] https://medium.com/james-reads-public-cloud-technology-blog/linux-tools-that-i-learned-10-years-ago-which-i-still-use-every-day-9289f952f169
[55] https://www.tecmint.com/install-ubuntu-alongside-with-windows-dual-boot/
[56] https://www.omgubuntu.co.uk/how-to-install-wsl2-on-windows-10
[57] https://docs.nvidia.com/cuda/wsl-user-guide/index.html
[58] https://developer.nvidia.com/cuda/wsl
[59] https://www.docker.com/blog/wsl-2-gpu-support-is-here/
[60] https://curl.se/

10.1 Andras resultat

Det är ett ganska öppet diskussionsklimat i Sverige där man gärna delar med sig av erfarenheterna för allas vårt bästa, Sverige och svenska är ett litet land och språk så vi behöver samarbeta för att komma någon vart eftersom det behövs stora mängder data och god datorkraft. Det finns några centra; AI Sweden[61], AI-innovation-sweden[62] och RISEs satsning på Språkteknologi[63]. Det har varit många intressanta seminarier där föreläsare generöst har delat med sig av kunnande och erfarenheter.

10.1.1 Arbetsförmedlingen

Har tagit fram en egen språkmodell men inte riktigt lika kraftfull som KBs. Den användes bl a för att matcha sökandes meriter och yrken till olika platsannonser (tekniken NER användes).

10.1.2 Skatteverket

Skatteverket har gjort en del för att hantera ärenden och mail. Dock var vi som lyssnade av säkerhetsskäl inte tillåtna att spela in presentationen. För översättningar kan t ex Skatteverket inte använda Google Translate lagligt av säkerhetsskäl då det körs på en server utanför Sverige.

10.1.3 Rise

Rise har som nämnt ovan gjort stora instsatser för att sprida kunskap, bl a via AI Sweden och lärorika föredrag inkl diskussioner varannan vecka. Många av dem är inspelade[64].

10.1.4 Peltarion

En av de stora småföretagen (ca 100 anställda) inom kommersialisering av AI där de t ex tillhandahåller gratis test av idéer under en provmånad i ett lättanvänt webb-gränssnitt där man inte behöver kunna programmera.

10.2 AI Sweden

Har bl a tagit fram språkmodeller[65] för svenska myndigheter.

[61] https://www.ai.se/en
[62] https://www.lindholmen.se/nyheter/nu-startar-ai-innovation-sweden
[63] https://www.ri.se/sv/vad-vi-gor/expertiser/sprakteknologi
[64] https://www.ai.se/en/projects-9/applied-language-technology
[65] https://www.ai.se/en/news/language-models-swedish-authorities

10.3 Swedish Language Data Lab

Swedish Language Data Lab[66] är Vinnova-finansierat och koordinerat som en av AI Swedens NLP hörnstenar. Inom projektet, som avslutades Maj, har man bl a tagit fram två svenska NER-modeller

10.4 Recorded Future

RF är ett kommersiellt (Amerikanskt?) företag som tillhandahåller några språkmodeller inom NER och försök att bedöma känsla (sentiment) i texter[67] Exempel på hur RF använder Sentiment på svenska finns här[68]

10.5 Språkbanken

Språkbanken Text har nyligen tagit fram nya resurser[69], bl a utvärderings- och analys mängderna Superlim för att utvärdera texter, detta kommer även att var användbart för att skapa nya språkmodeller och liknande. Det finns även Språkbanken Tal[70] som är en del av den nationella forskningsinfrastrukturen Nationella språkbanken där det i framtiden kommer finnas resurser för att omvandla tal till text-omvandlingar och omvänt samt möjligheter att tidslinjera text med talade sekvenser.

10.6 Användning av NLP inom hälsovård

Nyligen har det startats diskussionsgrupper med föredrag om hur man kan få in NLP inom hälsovården[71]. Förutom företag är det mest Region Halland resp Västergötland (via Sahlgrenska Science Park) som är aktiva[72] Ett av problemen är GDPR och anonymisering och det är inte helt lätt; om man t ex ersätter ett namn med ett annat så kan det ju bli lite konstigt om hemorten är i Somalia.

10.7 Några saker som är på gång

10.7.1 Electra

Ny bättre språkmodell för svenska är på gång.

[66] https://www.ai.se/en/node/81535/swedish-language-data-lab
[67] https://huggingface.co/RecordedFuture
[68] https://recordedfuture-analytics.medium.com/sentiment-analysis-in-swedish-24d439067824
[69] https://spraakbanken.gu.se/resurser
[70] https://sprakbanken.speech.kth.se/
[71] https://www.sahlgrenskasciencepark.se/events/pioneering-natural-language-processes-nlp-in-swedish-healthcare/
[72] https://sv-se.invajo.com/event/sahlgrenskasciencepark/pioneeringnaturallanguageprocessesnlpinswedishhealthcare

10.7.2 Andra typer av mer effektiva nät

Nya strukturer för att klara sig med mindre krävande hårdvara men ändå vara nästan lika bra. Nyligen presenterades en tänkbar efterföljare till Transformernät, Switch-transformern [FZS21] som kan tänkas öka prestandan trots mindre resurser.

10.8 Chatbottar

Det finns en uppsjö av chat-bottar och nu (september 2021) har det precis kommit ut nya möjligheter via NLP-teknik så intresserade uppmanas ha god koll på de senaste framstegen.

10.8.1 NeMo

Nvidia tar fram många modeller av olika slag, antagligen för att promota sin egen hårdvaruplatform. Ett exempel inom vårt område är NeMo[73] som innehåller programvara för hantering av bl a text.

Del IV
Implementation

11 Programmeringsmiljö

Vi har valt programmeringsspråket Python[74] för att det innehåller så mycket färdigt som gör att det blir snabbt att prova lite olika saker. Däremot kan det ta tid att köra programmen, typiskt för att tröska igenom ett större dokument (200 A3-sidor med text och bilder) för sammanfattning eller svar på frågor är ca 15 min med 32 GB RAM och en 8 kärnig CPU Intel(R) Core(TM) i7-9700K vid grundfrekvens 3.60 GHz med ett 11 GB RAM Nvidia RTX 2080 Ti under Windows 10. I den maskinen finns fläktar och vätskekylning som ändå låter högt under denna kvart så det är lämpligt att vara i ett annat rum.

Det finns många sätt att snabba upp detta, t ex genom att optimera koden och/eller använda varianterna pypi[75] eller cython[76]. Det går förstås att översätta till andra datorspråk, t ex C/C++ och/eller parallellisera för att få ytterligare prestanda men det kan vara tidsödande att få till.

11.1 Versionshantering

För att hantera olika versioner av programkoden är det lämpligt att använda github och i vårt fall har vi lagt delar av källkoden under Kalle Proroks använ-

[73] Nvidia. Developer Blog: NeMO, Juni 2021
[74] https://www.python.org/
[75] https://www.pypy.org/
[76] https://en.wikipedia.org/wiki/Cython.

dare nutte2 i[77] så att fler kan använda den och förhoppningsvis lära lite från den.

Det som kan vara lite besvärligt är också att både Python och programbiblioteken uppdateras så det kan vara ett bekymmer att kombinera ihop versioner som passar, normalt får man felmeddelanden, kanske först under körning om det inte är kompatibelt. För att hålla reda på detta är det lämpligt att ha en virtuell miljö[78] där man installera rätt kombination av versioner, ofta beskrivna i en requirements.txt-fil som man använda via kommandot pip[79] och alltså ha flera olika virtuella miljöer för sina olika program och versioner.

11.2 Utvecklingsmiljö

Med Python följer en redigerare (editor) IDLE (efter Eric Idle, en av medlemmarna i humorgruppen Monty Python) där man kan testköra och utveckla program. Den är dock alldeles för rudimentär och begränsad för lite större krångliga projekt, t ex saknas möjlighet att enkelt felsöka (debugga) sina program. Lämpligare alternativ kan vara gratisversionerna Visual Studio Code[80] eller PyCharm[81] som vi använt. Det finns också många andra bra miljöer, t ex (Ana)conda, Spyder mm.

11.3 Jupyter Notebook

Ett trevligt sätt att utveckla på är via en Jupyter notebook[82], i den kan man köra block med kod och även varva med instruerande text inkl bilder. En av fördelarna är att man också lätt kan ändra i något av blocken och sen köra om enbart denna del, det kan vara väldigt tidsbesparande ifall t ex något av de tidiga blocken gör något som tar lång tid, t ex installation av stora bibliotek eller hämtning av stora språkmodeller, nackdelarna kan vara att det kan bli svåröverskådligt om det är mycket kod samt att ingenting kan hållas hemligt (på något enkelt sätt).

11.4 Google Colaboratory

En variant av Jupyter notebooks är Googles snarlika on-line tjänst Colab[83] , med den kan man få tillgång till kraftfulla beräkningsgrafikkort men med vissa begränsningar i t ex tid (en timme), det finns även en PRO-version men den är för närvarande (aug 2021) inte åtkomlig i Sverige. Colab använder en variant av Linux och har flera förinstallerade bibliotek, t ex Tensorflow/Keras och Numpy så det går fort att komma igång och dessutom gratis.

[77] https://github.com/nutte2/TRFV
[78] https://docs.python.org/3/library/venv.html
[79] https://pip.pypa.io/en/stable/user_guide/
[80] https://code.visualstudio.com/
[81] https://www.jetbrains.com/pycharm/
[82] https://jupyter.org/
[83] https://colab.research.google.com/notebooks/intro.ipynb

12 Tack

Vi vill framföra vårt stora tack till Jonas B Jonsson på Trafikverket i Umeå, utan hans entusiasm och ledarförmåga hade detta projekt inte blivit av. Tack vare honom har vi tagit oss igenom tunga perioder, med hårdvarubrister och ibland dålig respons, tack vare Jonas goda omdöme, humör och humor. Tack också till Katarina Jonsson på Sweco för goda bidrag, framför allt gällande vattenanmälningar som vi hoppas kunna jobba vidare på närmaste tiden så resultatet av detta arbete blir verkligt användbart. Kalle vill också rikta ett fint tack till Frank Drewes för entusiasmerande, gott och trevligt samarbete! Frank har varit upptagen med många saker men ändå tagit sig tid till att hantera detta projekt på bästa sätt. Kalle riktar också ett stort tack till flickvännen Carina som fixat och grejat mycket hemmavid under tiden som möten, läsning, kodning, körning och felsökning pågått, jag hoppas kunna gengälda det nu framöver då arbetstiden förhoppningsvis minskar lite, sorry också för våra extrahöga elräkningar.

Frank vill i särskilt hög grad tacka Kalle för sin insats i projektet. Kalles tekniska kunnighet och hans vilja att söka kontakt samt dela med sig var nog den största tillgången det här projektet hade.

Sen vill vi alla förstås rikta ett stort tack till Trafikverket som gett oss förmånen och förtroendet att få jobba med detta, vi hoppas att resultatet kommer bli användbart och spara tid och pengar för liknande framtida projekt och må Norrbottniabanan bli ett riktigt bra projekt!

12.1 Adresser

Kalle kan nås på kalle.prorok@gmail.com eller 070-3 33 35 37.

Referenser

[CGG+21] Fredrik Carlsson, Amaru Cuba Gyllensten, Evangelia Gogoulou, Erik Ylipää Hellqvist, and Magnus Sahlgren. Semantic re-tuning with contrastive tension. In *9th International Conference on Learning Representations, ICLR 2021*, 2021.

[FZS21] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. https://arxiv.org/pdf/2101.03961.pdf, January 2021. Accessed:2021-06-18.

[MBH20] Martin Malmsten, Love Börjeson, and Chris Haffenden. Playing with words at the national library of sweden – making a swedish bert, 2020.