Luento 2: Kieli ja sen merkitykset

Sam Hardwick & Vadim Kulikov

13.12.2017 10:30-12:00

Perehdymme tekstianalyysin tehtäviin ja menetelmiin. Suomen kielen erityispiirteet ja esikäsittely. Korpus kertoo itsestään ja maailmasta. Keskeisenä esimerkkinä `word2vec` ja distributionaalinen semantiikka.

Kielen koneoppiminen

Koneoppiminen hyödyntää korpuksessa[1] olevaa implisiittistä ja eksplisiittistä tietoa[2], ja tätä tietoa voidaan käyttää korpuksen ymmärtämiseen ja sovelluksien rakentamiseen.

Aiemmin kielen ilmiöiden asiantuntijat ohjelmoivat tietokoneita jäsentämään kieltä kieliopin perusteella. Tämä oli monivaiheinen prosessi. Teksti jaettiin lauseiksi ja sanoiksi, yhdyssanat jaettiin osiin, sanojen perusmuodot selvitettiin, sanaluokat, taivutus ja johdokset analysoitiin, tehtiin syntaktinen ja semanttinen analyysi. Ihmisen näkökulmasta nämä ilmiöt ovat erillisiä ja jokaisessa on tarpeellista informaatiota.

Tilastollisten menetelmien ja koneoppimisen myötä suurta tekstimassaa itseään käytetään itsensä analysoimiseen. 90-luvulla ala koki "tilastollisen vallankumouksen", jossa analyysikerros kerrallaan ihmisten tekemät kieliopit korvattiin malleilla jotka oppivat esimerkeistä. Esimerkit ovat käytännössä tekstikokoelmia joissa tekstiin on ihmisten toimesta merkitty haluttu annotaatio. Kuluneen 5 vuoden aikana on tapahtunut uusi vallankumous, neuroverkkojen ja syväoppimisen vallankumous, jossa analyysikerrosten väliset rajat ovat kadonneet, tulokset ovat merkittävästi parantuneet ja aivan uudenlaiset tehtävät ovat tulleet mahdollisiksi.

Aiemmalla luennolla puhuttiin kuvien luokittelusta ja regressiomalleista. Myös tekstejä voi luokitella esimerkiksi aiheen mukaan, tai sen mukaan ovatko ne sävyltään myönteisiä, kielteisiä vai neutraaleja.[3]

Kielen ilmiöt ovat kuviin verrattuna hyvin relationaalisia[4] ja ne vaikuttavat pitkänkin matkan päähän[5]. Teksti on yksiulotteista siinä mielessä että se etenee vasemmalta oikealle, mutta sen rakenteisuuden takia se edellyttää erilaisia menetelmiä kuin kuvien tai yksinkertaisten aikasarjojen analysointi.

Valtaosa alan tutkimuksesta käyttää kohdekielenä englantia. Tällä on pari merkittävää seurausta. Ensiksi, skenaario jossa on käytettävissä laadukasta käsin luokiteltua opetusmateriaalia on ylikorostunut. Käytännössä on usein niin, että joko kieli tai tehtävä on sellainen ettei valmista opetusmateriaalia ole saatavilla[6]. Toiseksi, se mikä toimii englannin kohdalla ei välttämättä toimi jonkin toisen kielen kohdalla. Englannissa on melko luontevaa käyttää perusyksikköinä juoksevan

[1] Korpus on mikä tahansa tekstikokoelma, kuten sanomalehden vuosikerta tai 10 000 twiittiä.

[2] Esimerkiksi tämä teksti sisältää implisiittistä tietoa suomen kielestä ja eksplisiittistä tietoa tekstianalyysistä.

[3] Voit miettiä, millaisia ongelmia voisi tulla vastaan kun yrität tehdä tätä feedforward-neuroverkolla.

[4] On aivan eri asia, löikö Jussi Markoa vai Marko Jussia.

[5] Jos tekstissä mainitaan "Niinistö", pitää muistaa oliko tekstissä ollut puhe Villestä, Saulista vai Jussista.

[6] Yksinkertaisesti siitä syystä että kukaan ei ole käsin merkinnyt isoon tekstimäärään haluttua tietoa tai analyysiä.

kielen sanoja.[7] Substantiiveja ei taivuteta kuin monikon ja yksikön mukaan, verbejä vain muutamassa aikamuodossa (säännöllisissä verbeissä muotoja on vain kaksi). Suomessa taas on taivutuksen, johdosten ja yhdyssanojen takia valtava määrä sanamuotoja. Mikäli näitä muotoja ei normalisoida tai piirteistetä,[8] aineistosta tulee tarpeettoman harva[9] ja malli joutuu käyttämään kapasiteettiaan mallintamaan asioita joihin se ei välttämättä sovi.

[7] Mahdollisesti tokenisoituna niin että *isn't* jaetaan "sanoiksi" *is* ja *n't*.

[8] Normalisoida esim. palauttamalla sanat perusmuotoihin, piirteistetä esim. analysoimalla sanoista ulos niiden morfologia (taivutus).

[9] Tämä koskee erityisesti sanatason malleja, joille eri sanat ovat täysin erillisiä, sellaisetkin kuin "autolla" ja "autolle". Merkkipohjaiset mallit suoriutuvat paremmin.

Data ja mallit

Koska malleissa ei yleensä ole minkäänlaista valmista tietoa kielestä, jokainen koneoppimistehtävä joutuu oppimaan kielen tyhjästä.[10] Käsin tehtyjen opetusaineistojen[11] koko on mieluusti miljoonia sanoja, raakatekstin jota käytetään kielimallin[12] kehittämiseen[13] satoja miljoonia tai jopa miljardeja sanoja. Usein ollaan välimaastossa - käytetään sekä käsin merkittyä että raakatekstiä samassa mallissa[14] tai opitaan raakatekstistä kielen piirteitä[15] ja ohjelmoidaan edelleen niiden avulla varsinainen sovellus.

[10] Edellisessä kappaleessa mainittu esikäsittely on eräs tapa tuoda malliin valmista tietoa kielestä.

[11] "Supervised learning"

[12] "Language model"

[13] "Unsupervised learning"

[14] "Semi-supervised learning"

[15] "Feature learning"

Kielimalli

Kielimalli antaa todennäköisyyden jokaiselle merkkijonolle.[16] Se on siis todennäköisyysjakauma.[17] Yleensä se esiintyy yhteydessä jossa todennäköisyys on ehdollinen jo nähdystä. Esimerkiksi $P(\text{"kahvilassa"}|\text{"tapasimme"})$[18] on todennäköisyys että seuraava sana on "kahvilassa" kun tiedetään että sitä edelsi sana "tapasimme". Siihen vaikuttaa sekä kielen että maailman säännönmukaisuudet; $P(\text{"kahvilasta"}|\text{"tapasimme"}) < P(\text{"kahvilassa"}|\text{"tapasimme"})$ ja $P(\text{"Etelänavalla"}|\text{"tapasimme"}) < P(\text{"kahvilassa"}|\text{"tapasimme"})$.

Kielimallien kehitys on sinänsä oma akateeminen tutkimuskohteensa. Uusien menetelmien keskeisiin evaluointikriteereihin kuuluu se, miten matalan yllättyneisyyden[19] ne saavuttavat tyypillisillä vertailukorpuksilla.

Voi olla yllättävää että merkkijonojen todennäköisyyden laskeminen on niin tärkeää, mutta tätä ehkä selittää se, mitä kaikkea tällaiseen todennäköisyyteen liittyy. Jos tehdään kielimallia tavoitteena minimoida yllättyneisyys Wikipedian tekstiaineistossa, on hyödyksi tietää englannin kielen lisäksi kaikki tieto mitä Wikipediassa on. Koeta vaikka täydentää sellaisia lauseita kuin "_____ on Ranskan pääkaupunki" tai "aivo-selkäydinneste virtaa neljännestä aivokammiosta _____". Hyvin pärjääminen tällaisessa täydennystehtävässä vastaa sitä, miten paljon tietää maailmasta.

Toisaalta korpus kertoo myös kirjoittajastaan ja omista oletuksistaan. Nuoret käyttävät erilaisia sanoja kuin vanhemmat, naiset käyttävät miehiä enemmän adjektiiveja ja adverbeja, eri aikakausina ja eri maissa kirjoitetut tekstit ovat erilaisia. Nämä seikat ovat akateemisen

[16] Tai sanajonolle, tai muulle kielen osalle.

[17] ..eikä malli siinä mielessä kuin esimerkiksi neuroverkko on malli. Tässä on joskus sekaantumisen vaara.

[18] $|$ merkitsee ehdollista todennäköisyyttä. Tässä esimerkissä oli vain yhden edeltävän sanan konteksti, käytännössä todennäköisyys on ehdollinen suuremmasta kontekstista, mahdollisesti myös seuraavista sanoista; merkintätavat vaihtelevat.

[19] "Perplexity". Se on $2^{H(P)}$, missä $H(P)$ on kielimallin entropia yli annetun korpuksen. Entropia vastaa sitä, miten monta bittiä keskimäärin tarvitaan koodaamaan yksi sana, eli että miten yllättävä seuraava sana keskimäärin on kielimallille. Perpleksiteetin arvoa N voidaan tulkita niin, että malli on yhtä epävarma kuin jos se valitsisi N yhtä todennäköisen vaihtoehdon välillä. Luonnollisen kielen sanakohtainen perpleksiteetti on kertaluokkaa 100.

kielentutkimuksen kannalta kiinnostavia, mutta voi olla arvokasta myös yhteiskunnallisesta tai markkinatutkimuksen näkökulmasta pystyä luonnehtimaan ihmisiä ja heidän ajatuksiaan esimerkiksi Internetin keskustelupalstoilta löydettyjen tekstien perusteella.

Tekoälytutkimuksen vauhdittamiseksi onkin perustettu Marcus Hutterin toimesta kilpailu, Hutter Prize[20], jossa tehtävänä on pakata 100 megatavua tekstiä englanninkielisestä Wikipediasta mahdollisimman pieneen tilaan. Tämä on hyvin samankaltainen tehtävä kuin hyvän kielimallin kehittäminen (sillä rajoituksella että kielimalli pitäisi pystyä kuvaamaan tiiviisti).

[20] http://prize.hutter1.net/

Käytännössä kaikissa kieliteknologisissa sovelluksissa on implisiittinen tai eksplisiittinen kielimalli, siis sen mallin tai mallien lisäksi joilla kuvataan kiinnostuksen kohteena olevaa ilmiötä.

Neuroverkot

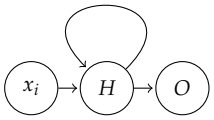
Kuvien luonnollinen representaatio on pikselitaulukko. Kun koulutetaan neuroverkkoa kuva-aineistolla, on yksinkertaista valita kuvien koko ja väriavaruus etukäteen[21] ja esikäsitellä materiaali sen mukaan.

[21] Esimerkiksi 100×100 pikseliä, väriavaruus $8 \times 8 \times 8$.

Tekstiä ei voi helposti esikäsitellä määrämittaiseksi. Jotta neuroverkko voi oppia tekstistä, teksti pitää joko piirteistää niin että kaikkien näytteiden representaatio on saman mittainen, tai käyttää sekvenssimallia joka pystyy käsittelemään mielivaltaisen pitkän näytteen, ja lopuksi erityisen lopetussymbolin.

Sekvenssimallin voi toteuttaa neuroverkolla, jossa yksi kerros saa syötteeksi sekä syötteen että oman tilansa edellisen syötteen jäljiltä. Siinä on siis sykli.[22] Kuten neuroverkoissa usein, lopputulos paranee kun kerroksia lisätään. Jos rekurrentteja piilokerroksia on kaksi, ensimmäinen saa syötteeksi alkuperäisen syötteen ja oman edellisen aktivaationsa, toinen saa ensimmäisen tason uusimman aktivaation ja oman vanhan aktivaationsa.

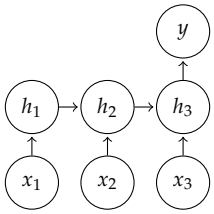
[22] Yksinkertaisen rekurrentin neuroverkon arkkitehtuuri voisi näyttää tältä:



Tässä x_i on järjestyksessä i:s syöte, H on yksittäinen *hidden layer*, O on output-kerros. Syöte on tyypillisesti aakkoston (tai sanaston, jos tehdään sanamalli) *one-hot*-vektori jossa yksi arvoista on 1 ja muut ovat 0.

Mitä sekvenssimalli sitten tuottaa - yhden luokituksen vai yhtä monta luokitusta kuin sekvenssissä on kirjaimia tai sanoja? Molemmat ovat mahdollisia. Syötesekvenssin päätteeksi voidaan antaa piilotilan aktivaatio *output*-kerrokselle joka tuottaa yhden luokan[23]. Tämä voisi olla esimerkiksi sentimentti, eli tekstin tunnesävy, tai binäärinen luokitus vaikkapa sen mukaan, käsitelläänkö tekstissä tietyn kaupallisen alan tuotteita.

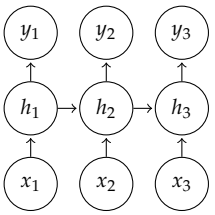
Output-kerros voidaan lukea myös jokaisen aktivaation kohdalla, jolloin jokaista syötemerkkiä vastaa yksi tuloste[24]. Vaativimmissa tehtävissä, kuten konekäännöksessä, käytetään usein *encoder-decoder*-mallia, jossa ensin käydään koko syöte läpi, saadaan tuloksena yksi lopullinen piilotilan aktivaatio, joka sitten syötetään syklisesti output-kerrokselle ("decoder") kunnes tuloksena on lopetussymboli sen merkiksi että tuloste on valmis.



[23] Kolmen syötemerkin sekvenssi tuottaa yhden tuloksen. h_i ovat saman piilotilan aktivaatioita kunkin syötteen jälkeen.

Sekvenssi-sekvenssi -mallissa voidaan lukea myös jokaisen aktivaation kohdalla, jolloin jokaista syötemerkkiä vastaa yksi tuloste[24]. Vaativimmissa tehtävissä, kuten konekäännöksessä, käytetään usein *encoder-decoder*-mallia, jossa ensin käydään koko syöte läpi, saadaan tuloksena yksi lopullinen piilotilan aktivaatio, joka sitten syötetään syklisesti output-kerrokselle ("decoder") kunnes tuloksena on lopetussymboli sen merkiksi että tuloste on valmis.

Sekvenssi-sekvenssi -mallit ovat potentiaalisesti erittäin ilmaisuvoimaisia, koska periaatteessa mitä tahansa rakenteista informaatiota voidaan koodata tekstimuotoon. Opetusaineistossa voi olla lähtöse-



[24] Sekvenssi-sekvenssi -malli

kvenssinä tekstiä, ja kohdesekvenssinä tekstiä jota on jotenkin annotoitu – esimerkiksi syntaktisesti, nimettyjen kohteiden mukaan[25], tai kohteena voi olla tekstistä pääteltävissä olevat relaatiot[26]. Lähteenä voi myös olla jotain muuta kuin kieltä. Esimerkiksi kuva voidaan enkoodata piilotilaan, josta sitten dekoodataan selostus siitä, mitä kuvassa on.

Sekvenssimallien toteuttamiseen ei yleensä riitä "tavalliset" *fully connected* -kerrokset tai konvoluutiot, koska niiden on vaikea oppia "muistamaan" pitkän aikavälin ilmiöitä. Tästä syystä niissä käytetään erityisiä muistineuroneita. Näistä ensimmäisenä läpimurron tehnyt ja edelleen paljon käytetty oli LSTM, *long short-term memory* (1997). LSTM-yksikkö on tavallaan oma pieni neuroverkkonsa, joka muistaa omassa sisäisessä aktivaatiovektorissaan aiemmin tapahtuneita aktivaatioita. Kun yleensä kahden neuronin välistä yhteyttä kuvaa yksi paino, LSTM-yksiköillä näitä on kolme. Yksi on tavanomainen input-aktivaatio, yksi "output gate" ja yksi "forget gate". *Output gate* säätelee sitä, pitäisikö LSTM-yksikön antaa vaste vai ei, ja *forget gate* sitä, pitäisikö LSTM-yksikön pyyhkiä muistinsa vai ei.

LSTM ja siitä kehitellyt variantit (jotka ovat yleensä yksinkertaisempia, esimerkiksi sellaisia joista puuttuu *forget gate*) ovat tässä kuvatuissa rekurrenteissa (*recurrent*) neuroverkoissa niin tavallisia, että yleensä RNN[27], *recurrent neural network*, tarkoittaa verkkoa jossa on vähintään yksi syklinen piilokerros jossa on muistineuroneita.

Aivan viime vuosien ilmiö on ollut syvät rekurrentit verkot, jotka tekevät suoraan raakatekstistä raakatekstiin erittäin vaativia tehtäviä ilman minkäänlaista esikäsittelyä tai jäsennystä. Tällaisia järjestelmiä jossa yksi jättimäinen verkko tekee kaiken kutsutaan nimellä *end-to-end*. Esimerkiksi Googlen monikielinen konekäännös toimii näin. Sama verkko kääntää kaikki kieliparit, ja syötteessä annetaan yksinkertaisesti erityissymboli joka kertoo dekoodauskerrokselle, mikä on haluttu kohdekieli. Piilokerrosten aktivaatiot ovat ikään kuin kielineutraali kuvaus siitä, mitä jokin teksti *tarkoittaa*, ja nämä sitten puretaan auki kohdekielen tapaan ilmaista tämä asia. Mukana on vielä *attention*-niminen osa, joka ohjaa dekoodausta antamaan suuren painon sellaisille osille syötettä joiden tiedetään olevan tärkeitä sille osalle tulosta jota ollaan juuri laskemassa. Googlen konekäännöksen voisi siis sanoa olevan *deep end-to-end encoder-decoder recurrent neural network with attention*!

Muistillisia neuroverkkoja on myös käytetty konvoluutiomalleissa, jolloin yleensä ei voida ottaa huomioon rajattoman suurta kontekstia.

Miten paljon dataa tarvitaan?

Ihannetapaus olisi se, että kutakin tehtävää varten olisi miljoonia esimerkkejä halutusta syötteestä ja tuloksesta. Vaativissa tehtävissä tämä vaatii kohtuuttoman paljon käsityötä, tai lähdemateriaalia saattaa yksinkertaisesti olla niukasti. Tällöin voidaan erikseen opettaa kielimalli (tähän riittää suuri määrä raakatekstiä, jota on helposti saatavilla) ja

[25] NER, *Named Entity Recognition*, merkitsee ja luokittelee tekstissä mainitut ihmiset, organisaatiot, tuotteet, ajankohdat ym.

[26] Kuka on kenenkin kanssa naimisissa, mikä tapahtuma oli minkäkin tapahtuman seuraus.

[27] Tai c-RNN, *character RNN*, joka nimenomaan kuvaa merkkijonoja merkkijonoiksi; m-RNN, *multimodal RNN* jolla kuvataan kuvia tai videoita tekstiksi; jne.

tehtäväspesifinen malli. Esimerkiksi Baidun kehittämä m-RNN joka tuottaa automaattisesti kuville sanallisia kuvauksia siitä, mitä ne esittävät, sisältää neljä osaa:

1. Semanttisen representaation kaikille ison tekstikorpuksen sanoille löytävä malli
2. RNN-malli kielen eri osien välisistä suhteista
3. CNN-malli joka representoi kuvia
4. Dekoodauskerros, joka on käytännössä output-kerros kolmelle edellämainitulle

Vain viimeinen osa tarvitsee esimerkkejä kuvista joille ihminen on tehnyt kuvauksen. Näitä oli käytössä joitain kymmeniä tuhansia.

Kun tehtävän kannalta labeloitua dataa on paljon saatavilla, neuroverkot pystyvät vaikka mihin. Kymmenetkin tuhannet esimerkit ovat kuitenkin lopulta aivan liian vähän.

Piirreoppiminen (feature learning)

Edellä mainittu m-RNN on esimerkki piirreoppimisesta. Neuroverkon tavoitteena ei ole välttämättä ratkaista ongelmaa suoraan, vaan tuottaa lähdeaineistosta ikään kuin jalostettu tiivistelmä, jota voidaan käyttää jatkokehittelyyn - joko seuraavan neuroverkon tai ihmisen toimesta. Syvennymme tähän ajatukseen sanavektoreiden kautta.

Sanasemantiikka ja word2vec

Jos ulkoavaruuden olento saisi käsiinsä meidän kirjallisuuttamme, sen olisi mahdotonta päätellä yksittäisistä sanoista niiden merkitystä. Sanalla "kuningas" ei ole mitään selvää yhteyttä kuninkuuden käsitteeseen. Sanan "kuningas" ympärillä kuitenkin esiintyy säännönmukaisesti samoja sanoja, kuten "kruunattiin" ja "Ranskan". Olento voisi huomata että samoissa ympäristöissä esiintyy myös sana "kuningatar", ja että vastaavanlaisia eroja on sanoilla "kreivi" ja "kreivitär".

Tällaisiin havaintoihin perustuu teoria siitä, että sanojen esiintymisympäristöt sisältävät tietoa niiden merkityksestä[28]. Selvästi jos kaksi sanaa ovat täydellisen synonyymisiä, ne voi vaihtaa toisiinsa missä tahansa kohdassa tekstiä. Niillä pitäisi siis olla samanlaiset ympäristöt. Mitä samankaltaisempia sanat ovat, sitä samankaltaisemmat ympäristöt.

Voidaan kuvitella jättimäinen taulukko[29], jossa jokaisen kieleen kuuluvan sanan kohdalle on merkitty, miten monta kertaa se esiintyi yhdessä kunkin muun sanaston sanan kanssa. Tällainen taulukko sisältää kaiken tiedon sanojen esiintymisympäristöistä, ja jokaiselle sanalle saadaan representaationa vektori, joka kertoo miten monta kertaa kunkin muun sanan kanssa se on esiintynyt. Vektorin ulotteisuus on sama kuin sanaston koko, eli kymmeniä tai satoja tuhansia.

[28] Tunnetaan kielitieteessä nimellä "distributional hypothesis"

[29] Tähän tapaan:

	kuningas	kruunu	kuningatar
kuningas	5	9	4
kruunu	9	1	2
kuningatar	4	2	3

Nyt sanaa "kuningas" kuvaisi vektori $[5, 9, 4]$, sanaa "kruunu" vektori $[9, 1, 2]$ ja sanaa "kuningatar" vektori $[4, 2, 3]$.

Tällaisten vektorien sisältämän informaation hyödyntäminen on ollut mielenkiinnon kohteena pitkään. Vuonna 2013 `word2vec`-niminen neuroverkko osoittautui erittäin toimivaksi ja vaikutusvaltaiseksi lähestymistavaksi.

`word2vec`in tavoitteena on oppia ennustamaan sanoja niiden ympäristöistä[30]. Sen oppimisaineistona on siis syötteenä luettelo[31] sanoja, joita lähiympäristössä esiintyi, ja tuloksena yksi sana. Esimerkiksi ympäristösanat voisivat olla [Ludvig, Ranskaa, hallinnut, neljästoista, on, pisimmin, valtionpäämies] ja haluttu sana "kuningas"[32].

`word2vec`in piilokerroksessa on käyttäjän valitsema määrä neuroneita (yleensä joitain satoja), ja sanaston kokoinen output-kerros. Kun opetus on saatu valmiiksi, input-kerroksen ja piilokerroksen väliset painot voidaan sellaisenaan käyttää kuvaamaan sanoja. Jokainen sana saa vektorirepresentaation, jonka ensimmäinen alkio on sanan ja ensimmäisen piiloneuronin välisen yhteyden paino, toinen sanan ja toisen piiloneuronin välisen yhteyden paino, ja niin edelleen.

Kun nämä vektorit tulkitaan geometrisesti, voidaan havaita oppimistehtävän tuottavan hämmästyttävän hyvän sanasemantiikan. Erityisesti erilaiset merkityssuhteet näkyvät vektoreiden välisissä suhteissa. Esimerkiksi vektori joka vie sanasta *kuningatar* sanaan *kuningas* on lähellä vektoria joka vie sanasta *nainen* sanaan *mies*. Lineaarialgebralla vapaamuotoisesti ilmaistuna, $\overline{kuningas} - \overline{mies} + \overline{nainen} \approx \overline{kuningatar}$. Nämä piirteet eivät koske vain kieltä, vaan myös tekstistä löytyvää informaatiota. Tietosanakirja- tai sanomalehtitekstiaineistosta opetettu vektorimalli tietää että $\overline{Pariisi} - \overline{Ranska} + \overline{Saksa} \approx \overline{Berliini}$.

Koska vektorimalli sijoittaa samamerkityksiset sanat lähekkäin, sen avulla voi rikastaa lyhyempiä tekstinäytteitä. Kuvitellaan esimerkiksi malli joka sisältää tietoja jostain ihmisestä: asuinpaikka, syntymävuosi, sukupuoli ja pieni valikoima sosiaalisen median kirjoituksia ja profiilitekstiä. Tehtävänä voisi olla ennustaa hänen kiinnostuksensa jostain tuotteesta, tai todennäköisyys että hän syyllistyy luottokorttipetokseen tulevan vuoden aikana. Nämä piirteet sijoitetaan yhdessä regressiomalliin. Pienessäkin määrässä tekstiä on luultavasti arvokasta informaatiota, mutta siihen ei voi sovittaa tekstimallia. Tekstistä löytyvien sanojen sijaan kannattaa ehkä antaa mallille piirteiksi niiden vektorirepresentaatiot, joiden laskemisessa on käytetty apuna suurta tausta-aineistoa.

Lähimerkityksisyys on vektorimalleissa laaja käsite. Voidaan esimerkiksi kysyä, mikä vihannes on eniten porkkanan kaltainen, tai kuka poliitikoista on eniten Margaret Thatcherin kaltainen[33].

[30] Tai toisinpäin, ympäristöstä niitä sanoja joissa ne esiintyivät.

[31] Käytännössä *one hot*-vektori, eli sanaston kokoinen vektori jossa ympäristön sanojen kohdalla on lukuarvo 1 ja muiden kohdalla 0.

[32] Alkuperäisessä tekstissä oli siis lause "kuningas Ludvig neljästoista on Ranskaa pisimmin hallinnut valtionpäämies"

[33] Tässä auttaa se, jos opetusmateriaaliin voidaan merkitä henkilöt niin että etu- ja sukunimet pysyvät yhdessä. sense2vec -niminen järjestelmä perustuu opetusaineiston esikäsittelyyn sanaluokkien, kiinteiden fraasien ja nimettyjen luokkien avulla.

Tekstin aihe ja sentimentti

Perinteisillä menetelmillä saatiin hyviä tuloksia dokumenttien luokittelussa. Tässä skenaariossa kokoelma erillisiä kohtalaisen pitkiä tekstejä – sanomalehtiartikkeleita, Internet-foorumikirjoituksia, poliitikkojen puheita – luokitellaan niiden aiheen mukaan. Mahdollisesti niin että ihminen ensin luokittelee jonkin verran dokumentteja ai-

heen mukaan, ja koneoppimisella luokitellaan loput saman jaottelun mukaan[34].

Täysin *unsupervised* -skenaariossa puhutaan *latent topic modelling*ista. Tällöin malli luokittelee tekstejä "sokkona" N tuntemattomaan kategoriaan.[35]

Tilanne on hankalampi kun tekstillä ei ole yhtä ainoaa aihetta, vaan sen eri osilla on omat osa-aiheensa. Tekstissä saatetaan käsitellä esimerkiksi ruoanlaittoa tai rikollisuutta vaikkei siinä esiintyisi sanoja "ruoanlaitto" tai "rikollisuus", ja vaikka itse tekstin aihe olisi *Sopranos* -tv-sarja.

Sama koskee tekstin sentimenttiä. Tekstissä voi olla myönteisiä ja kielteisiä osia, ja näillä osilla voi olla eri aiheet kuin koko tekstillä.

Kuten monessa kieliteknologian tehtävässä, RNN:t ovat sentimenttianalyysissä eturintamassa. RNN voi tuottaa tekstin jokaiselle merkille lukuarvon, joka kertoo missä mennään tunnesisällön kannalta. Samoin RNN voisi tunnistaa, missä kohdassa tekstin aihe muuttuu – tosin tietääksemme tätä ei ole vielä kovin onnistuneesti tehty, ehkä sopivan opetusaineiston puutteen vuoksi.

Opetusaineiston saaminen on ylipäätään ongelmallista muiden kielten kuin englannin kannalta. Tällöin voidaan yhdistää syvistä neuroverkoista saadut piirteet, kuten sanojen vektorirepresentaatiot, ja perinteisemmät kieliteknologian ja koneoppimisen menetelmät. Voidaan esimerkiksi kerätä laaja kokoelma positiivista ja negatiivista sentimenttiä edustavia sanoja ja käyttää niitä sääntöjen kirjoittamisessa. Erityisesti voidaan opettaa malli mahdollisimman relevantilla tekstillä. Joissain yhteyksissä "konservatiivinen" on kielteinen luonnehdinta, toisissa myönteinen.

Tällaisen analyysin lähtökohtana voi olla paljon hienosyisempikin erottelu kuin positiivisuus vastaan negatiivisuus. Itse asiassa mikä tahansa polaarinen[36] piirre sopii. Halpa ja kallis, korkealaatuinen ja heikkolaatuinen, ystävällinen ja tyly, tai aktiivinen ja passiivinen. Tällöin teksti kertoo toisaalta itsestään ("miten tämä teksti näitä asioita asettaa semanttiseen avaruuteen"), toisaalta maailmasta ("missä järjestyksessä automerkit ovat luotettavasta epäluotettavaan").

[34] Tämä on esimerkki *semi-supervised learning*ista

[35] Erittäin vaikutusvaltainen algoritmi tällä saralla on *Latent Dirichlet Allocation*, LDA.

[36] Jolla on jossain mielessä kaksi ääripäätä.