

LATVIJAS UNIVERSITĀTE  
DATORIKAS FAKULTĀTE

**DAUDZVALODĪGU JĒDZIENTELPU PIELIETOJUMS  
NODOMU NOTEIKŠANĀ**

MAĢISTRA DARBS

Autors: **Viktorija Leimane**

Studenta apliecības Nr.: v116047

Darba vadītājs: Dr.sc.comp. Kaspars Balodis

RĪGA 2022

## ANOTĀCIJA

Daudzvalodīga lietotāja nodomu noteikšana ir nozīmīga virtuālo asistentu darbībā, un klientu apkalpošanas automatizācija kļūst arvien aktuālāka. Ievades teksta virknes tiek attēlotas daudzdimensionālā vektoru telpā jeb jēdzientelpā, kuru izmanto nodomu klasifikācijas modeļi, lai piegādātu lietotājiem tiem nepieciešamo informāciju. Darbā tiks apmācīti dažādi mašīnmācīšanās modeļi un salīdzinātas dažādas pieejas, piemēram, ievades teksta attēlojums uz daudzvalodīgu tekstu korpusu apmācītas jēdzientelpas vai ievades teksta mašīntulkošana uz angļu valodu pirms jēdzientelpas izveides.

**Atslēgas vārdi:** daudzvalodīgas jēdzientelpas, nodomu noteikšana

## **ABSTRACT**

**Keywords:** keyword

## SATURS

APZĪMĒJUMU SARAKSTS .....	4
IEVADS .....	5
1 LITERATŪRAS APSKATS .....	6
1.1 Jēdzientelpa.....	6
IZMANTOTĀ LITERATŪRA UN AVOTI .....	10
PIELIKUMS .....	10

## **APZĪMĒJUMU SARAKSTS**

NLP (natural language processing) - dabisko valodu apstrāde

Jēdzientelpa – (word embeddings) vārdu vai frāžu attēlojums daudzdimensionālā vektoru telpā

Word2Vec – (word to vector) jēdzientelpas implementācija, kurā individuālus vārdus aizstāj daudzdimensionāli vektori

PCA (principal component analysis) - galveno komponentu analīze

## IEVADS

Arvien lielāku daļu tirgus pārņem pakalpojumu industrija (% ES), un pakalpojumi arvien biežāk tiek piedāvāti globāli/starptautiski. Tam ir nepieciešams lietotāju dzimtās valodas atbalsts gan valstu valodu regulējumam, gan tirgus nišas ieņemšanas/tirgus konkurences dēļ (% ES iedzīvotāju svarīgi saņemt pakalpojumu savā dzimtajā valodā).

Uzņēmumiem tas galvenokārt ir izdevīgi, jo ļauj samazināt personālizdevumus (% pakalpojumu nozares uzņēmuma izdevumu). Tas savukārt samazina barjeru dalībai/iekļūšanai starptautiskā tirgū, kas nozīmē lielāku konkurenci un piedāvāto pakalpojumu daudzveidību. Lietotājiem, kuru dzimto valodu pārvalda mazs cilvēku skaits kā tas ir, piemēram, latviešu valodā, ir pieejami pakalpojumi, kuru tulkojumus būtu ekonomiski nerentabli nodrošināt ar algotu profesionālu personālu.

Darbā apskatītā metode nodrošina automatizāciju divos veidos: virtuālais asistents aizvieto klientu apkalpošanas speciālistu daudzvalodīgs modelis aizvieto profesionālu tulkotāju.

Darbs ir sadalīts teorētiskajā un praktiskajā daļā. Teorētiskajā daļā ir īsi aprakstīti mūsdienu modeļi un pieejas. Praktiskajā daļā ir veikti eksperimenti ar mērķi pielietot daudzvalodīgus modeļus un salīdzināt tos ar esošiem risinājumiem.

# 1. LITERATŪRAS APSKATS

## 1.1. Jēdzientelpa

Jēdzientelpa ir vārdu vai frāžu attēlojums daudzdimensionālā vektoru telpā. Jēdzientelpas no teksta korpusa iegūst ar neironu tīkliem, kuri uztver kontekstu no tuvākajiem vārdiem tekstā. CBOW (Continuous Bag-of-Words) metodē neironu tīkls mēģina uzminēt esošo (vidējo) vārdu no  $n$  iepriekšējiem un  $n$  nākošajiem vārdiem. Procesu atkārtojot, vārdiem, kas bieži parādās vienā kontekstā, būs līdzīgi vektori. Pēc distributional hypothesis vārdi, kas atrodas līdzīgos kontekstos, ir ar līdzīgu nozīmi.

Atšķirībā no dabisko valodu apstrādes metodēm, kas katru vārdu uztver kā vienu atsevišķu vienību un tādēļ vienīgā iespējamā darbība ar vārdiem ir pārbaudīt vienādību, katras jēdzientelpas vektora vērtības ietekmē vārdi tiem apkārt (distributed representation) un būtībā jēdzientelpas uztver attiecības starp vārdiem. Rezultātā vārdam atbilstošais vektors satur semantisku un sintaktisku informāciju par vārdu. No tā izriet praktiskā implikācija - ar vektoriem var darīt lineāro algebru - saskaitīt, atņemt utml.

Cilvēkiem uztverama jēdzientelpu analogija ir krāsas nosaukums un tam atbilstošais vektors RGB krāsu modelī ar R, G un B koordinātēm no 0 līdz 255, piemēram,  $red = (255, 0, 0)$ . Ar krāsu jēdzientelpām ir iespējams veikt saskaitīšanu un atņemšanu, kam ir fizikāla nozīme.

Atrast tuvākās krāsas sarkanam.

```
closest(colors, colors['red'])
# red (229, 0, 0)
# fire engine red (254, 0, 2)
# bright red (255, 0, 13)
# tomato red (236, 45, 1)
# cherry red (247, 2, 42)
```

Operācijas ar vektoriem darbojas gan krāsu nosaukumiem semantiski, gan skaitliskiem vektoriem krāsu telpā. Piemēram, tuvākais vektors violeta un sarkana starpībai ir zils, kas atbilst cilvēku intuīcijai par RGB krāsām.

$$purple - red = blue$$

$$(126, 30, 156) - (229, 0, 0) = (-103, 30, 156)$$

```
closest(colors, subtractv(colors['purple'], colors['red']))
# cobalt blue (3, 10, 167)
# royal blue (5, 4, 170)
# darkish blue (1, 65, 130)
# true blue (1, 15, 204)
# royal (12, 23, 147)
```

Tā saskaitot zaļu un zilu rodas kaut kas pa vidu - tirkīzs.

$$blue + green = turquoise$$

$$(3, 67, 223) + (21, 176, 26) = (24, 243, 249)$$

```
closest(colors, addv(colors['blue'], colors['green']))
# bright turquoise (15, 254, 249)
# bright light blue (38, 247, 253)
# bright aqua (11, 249, 234)
# cyan (0, 255, 255)
# neon blue (4, 217, 255)
```

No vektoru operācijām var nolasīt secinājumus par semantiskajām attiecībām starp vārdiem, piemēram, rozā sarkanam ir tas pats, kas gaiši zils zilam.

$$pink - red + blue = lightblue$$

$$(255, 129, 192) - (229, 0, 0) + (3, 67, 223) = (29, 196, 415)$$

```
closest(colors, addv(subtractv(colors['pink'], colors['red']), colors['blue']))
# neon blue (4, 217, 255)
# bright sky blue (2, 204, 254)
# bright light blue (38, 247, 253)
# cyan (0, 255, 255)
# bright cyan (65, 253, 254)
```

Izrādās tādas pašas sakarības kādas ir krāsu nosaukumiem un to attēlojumiem krāsu telpā ir spēkā jebkuram vārdam (ref 1.1. tabula). Vārdi, kuri bieži atrodas līdzīgos kontekstos, ir tuvāki



nozīmē. Jēdzientelpas ietver gan sintaktiskas, gan semantiskas attiecības starp vārdiem. Jāuzsver, ka tādas semantiskas attiecības kā valsts-galvaspilsēta nav eksplicīti uzdotas, jēdzientelpu modelis tās ir novērojis tikai balstoties uz vārdu atrašanās vietām teksta korpusā. Iespēja trennēt modeli uz neanotētiem datiem kā šajā gadījumā samazina modeļa izmaksas valodām, kurās anotēti dati ir mazāk pieejami, un daudzkārt palielina potenciālās treniņu kopas apjomu, kas parasti ļauj sasniegt lielāku precizitāti.

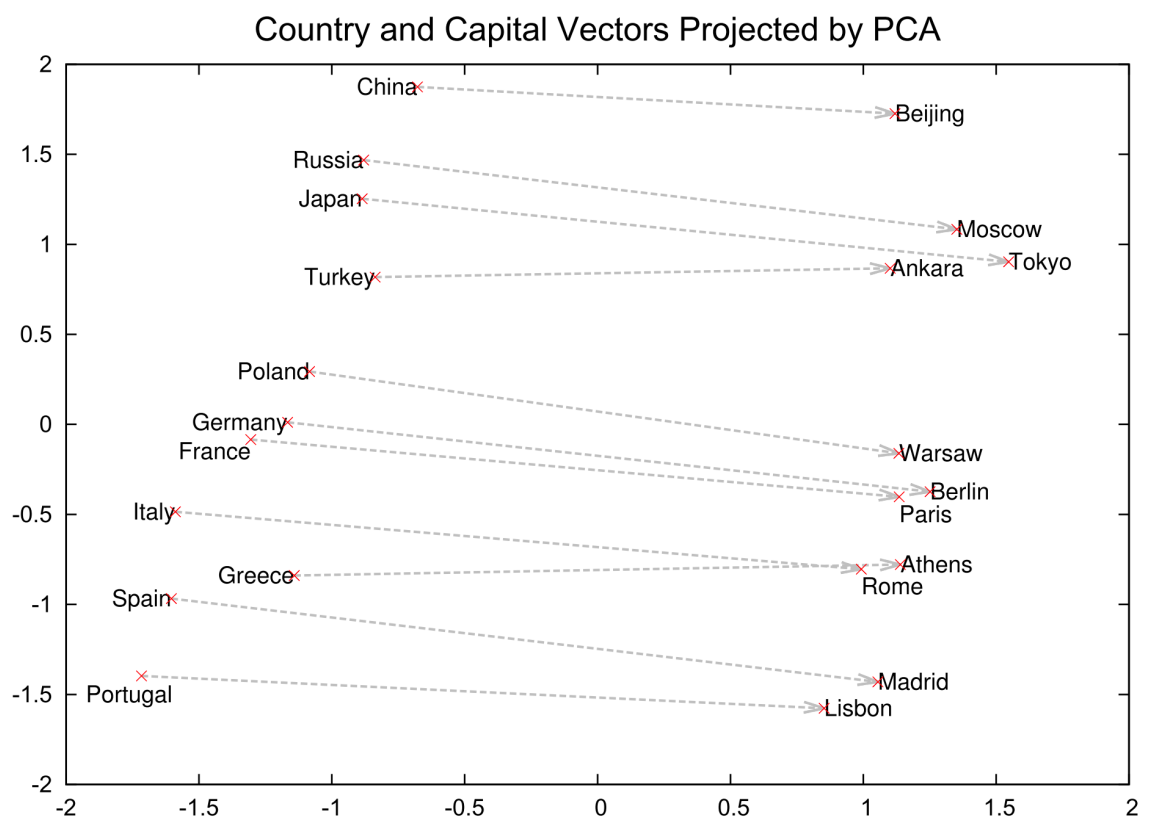
Spēja noteikt sintaktisku un semantisku vārdu attiecības ir īpaši būtiska virtuālo asistentu jomā, jo, pirmkārt, semantiski līdzīgiem nodomiem ir līdzīgi vektori, tātad tie vienādi klasificēsies, otrkārt, informācija par sintakses attiecībām noder, jo lietotāji ievada jautājumus brīvā formā un tas ir it īpaši svarīgi fleksīvām valodām kā latviešu.

1.1. tabula

Semantisko attiecību piemēri	
attiecība	piemērs
valsts-galvaspilsēta	Parīze - Francija + Itālija = Roma
valsts-valūta	dolāri - ASV + Latvija = eiro
vīrietis-sieviete	karalis - vīrietis + sieviete = karaliene

1.2. tabula

Sintaktisko attiecību piemēri	
attiecība	piemērs
daudzskaitlis	pele - peles
pagātne	staigā - staigāja
salīdzināmā pakāpe	labs - labāks



**1.1. att. Divdimensionāla PCA projekcija uzrāda attiecības starp valstu un galvaspilsētu jēdzien-  
telpām**

# PIELIKUMS

## Kods

Koda piemērs literatūras ievadā. Krāsu dati "xkcd.json" <https://github.com/dariusk/corpora/blob/master/data/colors/xkcd.json>.

Ideja <https://gist.github.com/aparrish/2f562e3737544cf29aaf1af30362f469>

```
import numpy as np
import json

def hex_to_int(s):
    s = s.lstrip("#")
    return int(s[:2], 16), int(s[2:4], 16), int(s[4:6], 16)

def distance(coord1, coord2):
    """Euclidean distance between two points
    """
    return np.sqrt(np.sum(np.subtract(coord1, coord2)**2))

def subtractv(coord1, coord2):
    """coord1 - coord2
    """
    return np.subtract(coord1, coord2)

def addv(coord1, coord2):
    """coord1 + coord2
    """
    return np.sum([coord1, coord2], axis=0)

def closest(space, coord, n=10):
    """as in https://gist.github.com/aparrish/2f562e3737544cf29aaf1af30362f469
    """
```

```
closest = []
for key in sorted(space.keys(),
                  key=lambda x: distance(coord, space[x]))[:n]:
    closest.append(key)
return closest

color_data = json.loads(open("xkcd.json").read())

colors = dict()
for item in color_data['colors']:
    colors[item["color"]] = hex_to_int(item["hex"])
```