Yleisesti

Loppuraportti

Loppuraportti on vapaamuotoinen kirjoitelma kurssista ja omasta oppimisprosessista. Raportista tulee kuitenkin ilmetä:

1. Mitä kurssiin sisältyi (minkälaisia aihealueita, minkälaisia tehtäviä jne.)
2. Kurssin suorittamiseen käytetty aika
3. Mitä opit
4. Mikä oli hankalaa ja mikä helppoa

Suoritan kurssin ohjeiden mukaisesesti tämän kurssin suorituksena yhden moodlesivuilta löytyvän kokonaisuuden. Olen valinnut kokonaisuudeksi:

Advanced NLP with spaCy - `https://course.spacy.io/en/`

Kurssissa on neljä eri osioita, reflektion osalta teen aina yhteenvedon kurssin ohjeiden mukaan jokaisen osion päätteeksi.

update: Kurssin tekemäni koodipätkät löytyvät nyt githubista - `https://github.com/heinohen/DIKI1001` repositoriosta.

1 Chapter: Finding words, phrases, names and concepts

Ensimmäisen osion aiheina on tutustuminen spaCy-kirjastoon ja tekstin prosessoinin alkeisiin.

Osio alkaa tutustumisella yleisesti mitä spaCy - kirjasto sisältää. Miten kirjasto ladataan ja muuta vastaavaa ihan perusjuttua. Näitä tuli jo harjoiteltua paljon edellisen periodin kurssilla DIKI1002 "Working text with python". Alkupuolella siis oli itselle lähinnä kertausta. Osiossa edetään sitten käyttämään jo koulutettuja pipelinejä joka sekin edelleen oli vanhan kertausta. Loppupuolella osiota päästiin uusiin juttuihin kuten annotaatioiden ennustamiseen, NERin käyttäminen kontekstissa ja sääntöpohjaisen Matcherin käyttö.

Tein osion loppupuolen tehtävät muutamaankin kertaan, jotta sai jäämään päähän mitä ja miten tässä prosessi etenee. Todella mielenkiintoinen osio loppupuolen osalta. Aikaa teorian tutkimiseen, lukemiseen ja tehtävien tekemiseen käytin "yhden illan"eli arvoilta noin 4-5 tuntia.

2 Chapter: Large-scale data analysis with spaCy

Mielenkiintoinen luku ja uutta asiaakin pääsi tekemään. Aikaa käytin tässä luvussa noin "kolme iltaa"eli about 7-8 tuntia. Teen tehtävät vscodella lokaalisti, saan oppina siitä enemmän kun pystyn debuggaamaan kunnolla. En muutenkaan ole kovin suuri selainohjelmoinnin ystävä. Harjoitukset olivat mielekkäitä ja onnistuin saamaan oikeat vastaukset ilman vinkkien tai vastausten tarkistusta. Ainoastaan tuo similarity - osio jäi ihmetyttämään kun kahdella eri koneella samalla tavalla koodaten sain identtiset mutta sivustolla eroavaisuuksia. Ladattu kielimallikin oli sama "medium", tähän kulutin muutaman tunnin ekstraa kun koitin järkeillä.

Omat:

```
import spacy
nlp = spacy.load("en_core_web_md")
# Compare two documents
doc1 = nlp("I like fast food")
doc2 = nlp("I like pizza")
print(doc1.similarity(doc2)) # 0.869833325851152
# compare two tokens
doc = nlp("I like pizza and pasta")
token1 = doc[2]
token2 = doc[4]
print(token1.similarity(token2)) # 0.685019850730896
#compare a document with a token
doc = nlp("I like pizza")
token = nlp("soap")[0]
print(doc.similarity(token)) # 0.18213694934365615
# Compare a span with a document
span = nlp("I like pizza and pasta")[2:5]
doc = nlp("McDonalds sells burgers")
print(span.similarity(doc)) # 0.47190033157126826
```

Ja esimerkit

```
# Load a larger pipeline with vectors
nlp = spacy.load("en_core_web_md")
# Compare two documents
doc1 = nlp("I like fast food")
doc2 = nlp("I like pizza")
print(doc1.similarity(doc2)) #0.8627204117787385
# Compare two tokens
doc = nlp("I like pizza and pasta")
token1 = doc[2]
token2 = doc[4]
print(token1.similarity(token2)) #0.7369546
# Compare a document with a token
doc = nlp("I like pizza")
token = nlp("soap")[0]
print(doc.similarity(token)) #0.32531983166759537
# Compare a span with a document
span = nlp("I like pizza and pasta")[2:5]
doc = nlp("McDonalds sells burgers")
print(span.similarity(doc)) #0.619909235817623
```

3 Chapter: Processing pipelines

Tässä luvussa pääsi luvun nimenkin mukaisesti suoritusjärjestystä ja luomaan omia komponentteja joille pystyi kirjoittamaan metodeja. Tämä luku oli selkeästi edellistä helpompi itselle koska ei keskittynyt niin lingvistiikan puolelle vaan enemmän koodaukseen pohjautuvaa tietotaitoa. Tämän osion kesto oli noin 2 iltaa koodausta ja sitten tämä reflektio-osuus tunnin - pari, eli sellainen 6-7 tuntia.

Mielenkiintoista oli kun osion loppupuolella näytettin hyviä ja huonoja tapoja tehdä koodia skaalaus ja suorituskyky mielessä pitäen. Tähän liittyen oli hyvä huomata, että pipelinen komponentteja saa disabloitua ja ajettua vaikka pelkästään tokenizeriä.