# Kodutöö 7 projekt: Pildifailist teksti tuvastamine ja sellest lühikokkuvõtte tegemine

## Töö eesmärk ja tehisintellekti kasutamise vajadus

Töö eesmärk oli luua programm mis saab sisendina ette pildifaili, millelt tuvastatakse tekst ning tekstist tehakse etteantud lausete arvuga kokkuvõte. Programm on mõeldud tuvastama ja töötlema eestikeelseid tekste.

Tehisintellekti on siin töös vaja:

1. Tuvastada pildilt eestikeelset teksti ja saada see sõnekujule – head tulemust ilma tehisintellektita ja masinõppemudeliteta on raske saavutada.
2. Omistada tuvastatud sõnadele-lausetele vastavad *embedding*'ud ehk vektoresitused ning leida nende omavahelisi tähenduslikke sarnasusi – sõnade, lausete tähenduste töötlemine on samuti selgelt tehisintellekti ja loomuliku keele töötluse lahendustega tehtav.

## Kasutatud ideed (nii kursuse materjalidest kui mujalt) koos viidetega

- Optiline pildituvastus
    - Pytesseract https://pypi.org/project/pytesseract/
- Loomuliku keele töötlus
    - EstBERT https://huggingface.co/tartuNLP/EstBERT/tree/main
    - SentenceTransformers https://www.sbert.net/docs/quickstart.html#comparing-sentence-similarities
    - Multilingual mudel SentenceTransformerile lause embeddingute leidmiseks https://www.sbert.net/docs/pretrained_models.html#multi-lingual-models
    - Klasterdamiseks k-means https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

    - Lausete eraldamiseks EstNLTK https://estnltk.github.io/

**Iga autori enda panuse kirjeldus**

- Joonas Tiitson - Koostas GUI ning ühendas selle tehisintellekti mudelitega. Selgitas välja kuidas Windows arvutis lokaalselt jooksutada Tesseracti ning estnltk-d.
- Kennar Kahju - Colabi töölehel pildilt teksti tuvastamine, kogu tekstiga kõige sarnasemate lausete algoritmi implementeerimine, exe faili koostamine lihtsamaks kasutamiseks
- Mattias Kimst - Colabi töölehel embeddingute leidmine, klasterdamisalgoritm, kõige üheks tervikuks vormistamine töölehel, testimine, töö kirjeldus

**Programmi testimisvõimalused ja vähemalt mõned testimistulemused**

Programmi kokkuvõtte tegemise funktsionaalsust testisime eraldi Jupyter Notebookis kahe tekstilõiguga, üks oli Tehisintellekti kursuse Moodles olev tutvustav tekst ja teine ERR-ist võetud uudis, selle võrdluse eesmärk oli võrrelda kahe implementeeritud algoritmi tööd, milliseid lauseid need välja valivad, kõikide algoritmide väljavalitud lausete hulgas oli kattuvusi teistega ehk võib oletada (aga mitte veel kindel olla), et kõik algoritmid lähtuvad lausete tähendustest ja ei vali lauseid juhuslikult.

Subjektiivsel hinnangul tundub 1. algoritm ehk sarnasus kogu teksti embeddinguga, kus embeddingud olid saadud EstBERT ja mean pooling abil, kõige paremini kokkuvõtvaid tekste tegevat. Nt Tehisintellekti kursuse tutvustusest Moodles tegi see järgmise kokkuvõtte, kust olid kenasti välja valitud kõige olulisemad laused ja laused stiilis “Kas arvuti suudab mängida "7 minutes in heaven" inimestest paremini?” kenasti välja jäetud:

```
Tutvuda saab nii teoreetilise küljega ( probleemid , lahendused , algoritmid ) kui ka praktiliselt , lahendades tehisintellekti ülesandeid praktikumides ning kodutöö jooksul .

Kursuse eesmärk on tutvuda tehisintellektiga ning mõnede selle valdkonna harudega nagu masinõpe , loomuliku keele töötlus , pilditöötlus jt.
```

Siit Moodle'st leiate loengute , prakside ja kodutööde materjale , ning siin saab vastata loengujärgsetele küsimustele , esitada koduülesannete ning eksami lahendusi ja vaadata oma hindeid .

Head ja konkreetset mõõdikut kokkuvõtte headuse hindamiseks ei tundu olevat peale subjektiivse hinnangu, kuna lausete semantilised tähendused ei ole alati ka inimesele endale lihtsasti ja üheselt määratavad/hinnatavad.

Jupyter Notebook lõpus on ühe korra testitud ka kogu protsessi, st tekstiga pildi etteandmist ja erinevate loodud algoritmide väljakutsumist sellel pildil, tervikuna kõik töötab. Ka on võimalik kogu lahendust tervikuna testida läbi graafilise kasutajaliidese laadides seal pilte üles ja valides kasutatavad algoritmid.

**Töö käigu kirjeldus: millised olid probleemid, mis õnnestus, mis jäi realiseerimata jne (kirjeldatud probleemid ega puudused ei mõju hindele kuidagi negatiivselt)**

Kõige suurema osa tööst moodustas info otsimine, kuidas kokkuvõtte tegemist tehniliselt lahendada - varem keegi meist ei olnud tuttav vastavate lahendustega. Samuti tuli välja sõeluda need lahendused, mida me saame eesti keele puhul kasutada, inglise keeles on palju teeke ja mudeleid, millega kokkuvõtteid teha, spetsiaalselt eesti keelele mõeldud analooge meil ei õnnestunud leida. Seetõttu kasutasime töötavat, aga väidetavalt mitte parimat meetodit ehk EstBERT embeddingute põhjal lõime lausete embeddingud (täpsem info, miks see parim ei ole on töölehel).

Võrdluseks lõime embeddingud ka multi-lingual mudeliga, mille internetist leidsime ja mis dokumentatsiooni kohaselt sobib ka eesti keelele (lisaks veel 50+ keelele). Seda lähenemist ehk konkreetselt lause embeddingute saamiseks loodud mudeli kasutamist peetakse paremaks lähenemiseks, kuid probleem on, et me ei leidnud, et spetsiaalselt eesti keelele seesugust valmismudelit leiduks, mis kahtlemata annaks paremaid tulemusi. Ise treenimiseks, peenhäälestamiseks on raskusi sobivate andmete leidmisega, ühe leitud juhendi järgi oleks vaja selleks lausepaare ja nende lausete vaheliste sarnasuste hinnanguid https://www.sbert.net/examples/training/datasets/README.html.

Saadud embeddinguid ühe lähenemisena klasterdasime k-means abil ja valisime igast klastrist kõige rohkem klastri keskel oleva, idee selleks saime uurides bert-extractive-summarizer valmisteegi lähtekoodi, mis ise eesti keelt ei toeta.

Teise lähenemisena proovisime leida väljaarvutatud kogu teksti embeddingule kõige sarnasemaid lauseembeddinguid. Mõlema lähenemisega leitud embeddingute põhjal loodud kokkuvõtetes oli kattuvusi, seega võib oletada, et need väga kardinaalselt erinevaid embeddinguid ei tagasta.

Pilditöötluse ehk teksti tuvastamisega läksime lihtsamat teed ja kasutasime Pytesseract valmisteeki ja eestikeelseid treeningandmeid Tesseractil, sest ise tehes arvatavasti ei oleks piisavalt hästi tuvastavat mudelit saanud. Samuti oli sisseloetud tekst vaja lausestada igale lausele vastava embeddingu leidmiseks, milleks kasutasime eesti keelele sobilikku teeki EstNLTK.

Kasutusmugavust silmas pidades lõime ka lihtsa kasutajaliidese, kust saab pilte üles laadida, valida lausete arvu kokkuvõttes ja kasutatavat algoritmi kokkuvõtte tegemisel. Kõige suuremad probleemid oli tesseracti ja programmi tööle saamisega kohalikus arvutis. Kuna colab ei lase teha kasutajaliideseid oma süsteemis. Selleks pidi kasutama Python 3.11 versiooni ning installima ja seadistama tesseracti asukoha. Samuti tessdata kausta süsteemi muutujates.