TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Marko Peedosk

Eestikeele sisukokkuvõtja edasiarendamine

Bakalaureusetöö (9 EAP)

Juhendaja:
Sven Aller, MSc

Tartu 2025

Eestikeele sisukokkuvõtja edasiarendamine

Lühikokkuvõte:

Infoküllases keskkonnas aitab automaatne sisukokkuvõtja säästa aega, tõstes esile olulisemad tekstiosad ja toetades kiiret orienteerumist sisus. EstSumi täiustati semantilise analüüsi võimalustega: lisati kaks lingvistilist moodulit ning katsetati siirdeõpet *few-shot* meetodil, et hinnata selle sobivust väikese andmestikuga töötamiseks. Üks moodulitest põhines wordnetil, parandades küll veidi täpsust, kuid vähendades süsteemi jõudlust märgatavalt. Keelemudelil põhinev moodul parandas täpsust ja säilitas töökiiruse. Siirdeõppe tulemuslikkus jäi samale tasemele EstSumi baasversiooniga. Keelemudelitel põhinevat lähenemisviisi saab edasi arendada, andes mudelile siirdeõppe käigus rohkem treeningandmeid ning täpsustades viipa. Lisaks loodi veebipõhine kasutajaliides, mis võimaldab süsteemi kasutada ka mittetehnilistel kasutajatel.

Võtmesõnad: EstSum, sisukokkuvõtja, tehisintellekt, WordNet

CERCS: P175 Informaatika, süsteemiteooria

Continued Development of an Estonian Text Tummarizer

Abstract:

In an information-rich environment, automatic summarization helps save time by highlighting key content and enabling faster text navigation. EstSum was enhanced with semantic analysis: two linguistic modules were added, and transfer learning was tested using a few-shot approach to evaluate performance with limited training data. The wordnet-based module slightly improved accuracy but significantly slowed the system, while the language model–based module achieved a negligible increase in accuracy without affecting speed. Although few-shot performance matched the baseline, further improvements are possible by supplying more training examples and refining prompts. To improve accessibility, a web-based user interface was also developed for non-technical users.

Keywords: EstSum, summarization, artificial intelligence, WordNet

CERCS: P175 Informatics, systems theory

Sisukord

1. Sissejuhatus

Tänapäeval seisavad inimesed silmitsi info ülekoormusega, kuna tekstide rohkus kasvab pidevalt. Viimasel ajal on olukord muutunud veelgi keerukamaks, kuna generatiivsete transformerite (ingl *generative transformers*) kasutuselevõtt on oluliselt suurendanud tekstitoodangu hulka. Selle tulemusena võib olla keeruline otsustada, millised tekstid väärivad süvenemist. Efektiivne ajakasutus on aga üha olulisem – seetõttu on suurenenud vajadus tööriistade järele, mis aitaksid inimestel oma aega säästa ja keskenduda vaid kõige olulisemale sisule. Üldjuhul moodustab ainult väike osa tekstis sisalduvast infost tegelikult väärtusliku või vajaliku sisu, mistõttu aitaks ainult olulise info esiletõstmine oluliselt vähendada lugeja ajakulu.

Lahenduseks on automaatsed sisukokkuvõtted, mis aitavad lugejal kiiresti mõista, millest tekst räägib ja kas see vastab tema huvidele või vajadustele. Sisukokkuvõtete genereerimist on uuritud juba alates 1958. aastast, mil Hans Peter Luhn ja hiljem Edmundson tutvustasid esimesi meetodeid [1] [2]. Käesolev töö keskendub eestikeelse sisukokkuvõtja EstSum edasiarendamisele – selle algatasid Tartu Ülikoolis Kaili Müürisepp ja Pilleriin Mutso [3]. EstSum tugineb Edmundsoni põhimõtetel, mille kohaselt hinnatakse lausete olulisust asukoha, sõnade sageduste ja võtmesõnade põhjal [2]. Käesolev arendus tugineb Janar Saksa varasemale tööle, milles EstSumiga ühendati lingvistiline moodul lemmatiseerimiseks ja stopp-sõnade loend eesmärgiga parandada süsteemi tulemuslikkust [4].

Selle töö eesmärk on esiteks luua sisukokkuvõtjale EstSum veebipõhine kasutajaliides, et ligipääsetavamaks ja kasutajasõbralikumaks kokkuvõtete loomise muuta. Lisaks veebirakendus on effektiivsem, kui et iga kasutaja paigaldab tarkvara oma masinale. Teiseks on eesmärk katsetada erinevate lingvistiliste moodulite lisamist, et parandada süsteemi täpsust ja sisulist asjakohasust. Viimaks hinnatakse, kuidas tehtud muudatused mõjutavad kokkuvõtete kvaliteeti ning millised komponendid annavad suurima panuse tulemuslikkuse paranemisse.

Käesolev töö jaguneb viieks peatükiks. Esimeses peatükis selgitatakse, mis on automaatsed sisukokkuvõtjad ning tutvustatakse lähenemisviise, mida nende arendamisel kasutatakse. Teine peatükk annab ülevaate Tartu Ülikooli loodud eestikeelsete artiklite sisukokkuvõtjast EstSum, kirjeldades selle struktuuri ja senist arengulugu. Kolmandas peatükis keskendutakse metoodikale: kirjeldatakse, millised lingvistilised moodulid EstSumile lisati ning kuidas need implementeeriti. Neljas peatükk esitab lisatud moodulite katsetamise tulemused ning sisaldab arutelu nende kohta. Viiendas peatükis käsitletakse EstSumile loodud veebipõhise kasutajaliidese arendust.

2. Ülevaade sisukokkuvõtjatest

Sisukokkuvõtja on tööriist, mille eesmärk on lühendada algset teksti viisil, mis säilitab selle kõige olulisema sisu, võimaldades lugejal kiiresti mõista, millest tekst räägib. Automaatne sisukokkuvõtete genereerimine on teadusvaldkond, mille algus ulatub aastasse 1958, mil Hans Peter Luhn [1] avaldas esimese teadustöö sel teemal. Sellest ajast alates on välja töötatud mitmesuguseid lähenemisi alates lihtsatest sageduspõhistest meetoditest kuni keerukate sügavõppemudeliteni, mis suudavad teksti sisu semantiliselt analüüsida ja ümber sõnastada.

Sisukokkuvõtjad kasutavad loomuliku keele töötluse (*NLP*) meetodeid, et koostada lühendatud tekst, mis esitab olulise teabe ühest või mitmest sisenddokumendist. Vaishal jt [5], Mehta ja Majumder [6] ning IBM [7] eristavad kahte peamist lähenemisviisi sisukokkuvõtte koostamiseks:

- Väljavõtte meetod (ingl *extraction*)
- Ülevaate meetod (ingl *abstraction*)

Kombineerides neid kahte saab ka rakendada nii-nimelist hübriidmeetodit. Neid meetodeid saab omakorda rakendada kahel viisil:

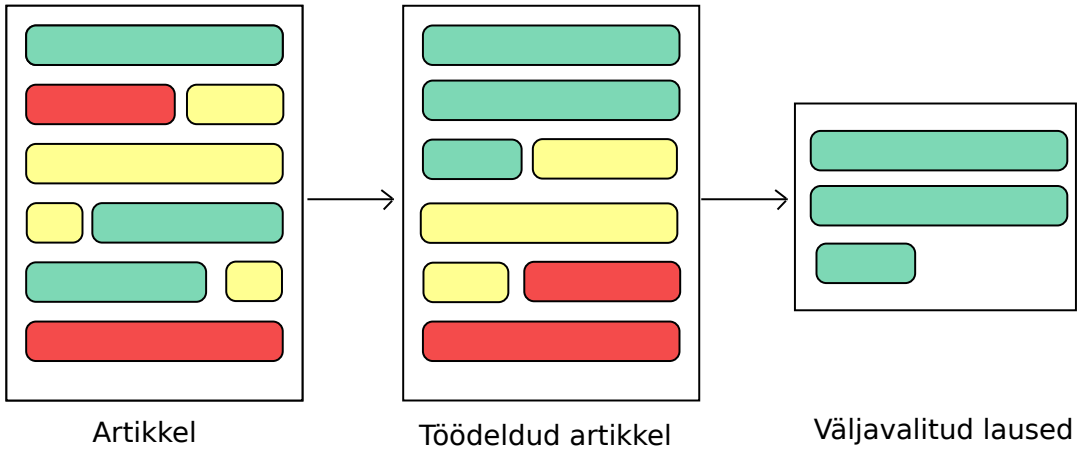
- Juhendatud õppega (ingl *supervised learning*);
- Juhendamata õppega (ingl *unsupervised learning*).

Järgnevalt käsitletakse lähenemisviiside tööpõhimõtteid ja omadusi.

2.1 Väljavõtte meetod

Mehta jt [6] ning IBM [7] sõnul keskendub väljavõtte meetod algdokumendist muutmata kujul lausete välja noppimisele. Seda illustreerib joonis 1. Vaishali jt [5] märgivad, et esmalt puhastatakse tekst, eemaldades kirjavahemärgid ja muu üleliigse ning jagades teksti üksikuteks sõnedeks. Seejärel arvutatakse igale lausele kaal, mis näitab selle olulisust. Lõpuks valitakse suurima kaaluga laused ning need ühendatakse kokkuvõtteks [5].

Joshi jt [8] toovad esile selle meetodi eelised: see on kiire, lihtne arendada ning tagab täpsuse. Täpsus selles mõttes, et loodud kokkuvõttes esinevad laused on sõnasõnaliselt täpselt samad, kui algdokumendis. Samas juhivad nad tähelepanu ka puudustele: kokkuvõtted võivad olla raskesti loetavad, kuna esineb korduvaid sõnu ja mõtteid, ning tekst võib jääda sidususe osas nõrgaks [8].



Joonis 1. Väljavõtte meetodi põhimõte, pärit raamatust[6].

2.2 Ülevaate meetod

S. Gupta ja S. K. Gupta [9] sõnul põhineb ülevaate meetod uute lausete loomisel, et edastada algse teksti tähenduse lühendatud kujul. See lähenemine eeldab teksti sügavamat mõistmist ning võimaldab luua kokkuvõtteid, mis on sujuvamad, sidusamad ja sageli loetavamad kui väljavõtte meetodite tulemused.

Seda saab üldjoontes jagada kolme kategooriasse [9]:

- **Struktuuripõhised lähenemised** – kasutavad eeldefineeritud struktuure, näiteks graafe, puid või malle, et korraldada ja esitada olulist teavet tekstist.
- **Semantilised lähenemised** – keskenduvad teksti tähenduse modelleerimisele, luues semantilisi esitusi, mille põhjal saab genereerida lühendatud sisu.
- **Sügavõppemudelid** – kasutavad närvivõrke, eelkõige järjestikuse sisendi ja väljundi (*seq2seq*) mudeleid, et õppida tekstide struktuuri ja tähendust ning toota uusi semantiliselt täpseid lauseid.

Ülevaate meetodi rakendamine on keerukas, kuna see eeldab mitte ainult keele grammatika mõistmist, vaid ka sisulist arusaama. Viimastel aastatel on selle valdkonna areng märgatavalt kiirenenud, eriti koos transformermudelite, nagu GPT, kasutuselevõtuga, mis on võimaldanud saavutada märkimisväärseid edusamme sisukokkuvõtete kvaliteedis ja usaldusväärsuses.

3. Lühiülevaade EstSumist

EstSum on Tartu Ülikooli poolt välja töötatud eesti keele sisukokkuvõtja, mis on peamiselt suunatud uudiste ja ajaleheartiklite kokkuvõtete loomisele [3]. Programm on kirjutatud programmeerimiskeeles Python, mis võimaldab tõhusat edasiarendamist, ning selle arhitektuur koosneb kolmest põhikomponendist [4]:

- HTML-muundur, mis teisendab sisendi SGML-vormingusse, eemaldades ebavajalikud märgendid ja lisades vajalikud (vt lisa V);
- Lausestaja, mis jaotab sisendi lauseteks vastavalt reeglitele, et tagada korrektne tekstistruktuur;
- Lausete väljavalija, mis määrab igale lausele kaalu ning koostab kokkuvõtte, lähtudes nende kaaludest.

Janar Saks [4] rõhutab, et iga lause kaal arvutatakse kolme teguri alusel: asukoha parameeter (P), formaadi parameeter (F) ja sageduste parameeter (K). Lause s kaal määratakse valemiga 1.

$$W(s) = \alpha P(s) + \beta F(s) + \gamma K(s), \tag{1}$$

kus α, β, γ on parameetrite kaalud vahemikus 0 - 1.

Lisaks kasutab Saks [4] süsteemis lemmatiseerimist, et sama tüvega sõnavormid (nt mets, metsa, metsas) koondada ühtsesse algvormi. See võimaldab käsitleda erinevaid vorme ühe võtmesõnana, vähendades mõju, mida kirjapildi erinevused kaalule avaldaksid. Samuti kasutatakse stoppsõnade filtreerimist, et kõrvaldada sageli esinevad, kuid sisulise analüüsi seisukohalt ebaolulised sõnad nagu ’olema’, ’või’ ja ’ja’ [4].

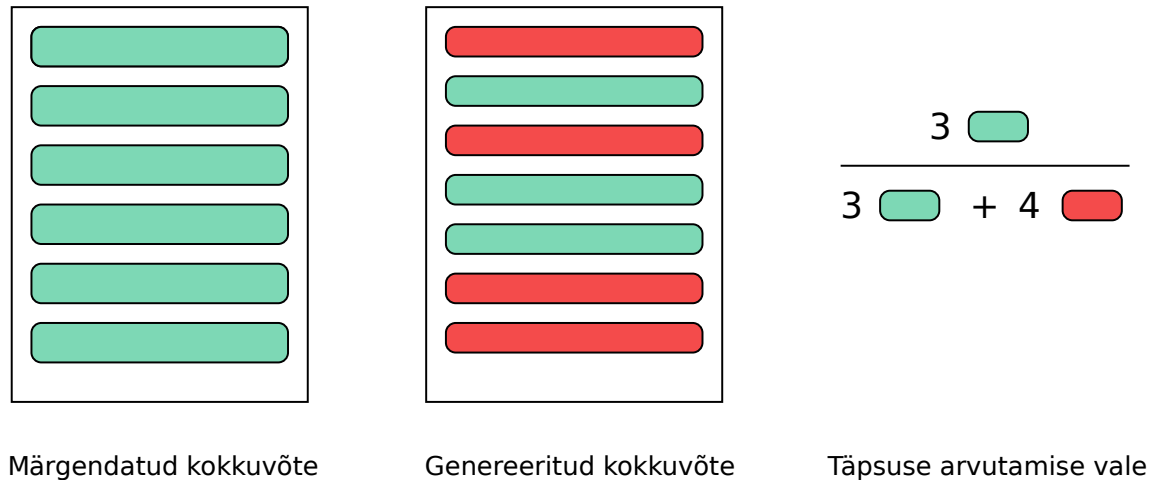
4. Metoodika

Selle peatüki eesmärk on kirjeldada meetodeid ja tehnikaid, mida kasutati EstSum täiustamisel. Töö käigus keskenduti kahele peamisele lähenemisele tulemuslikkuse parandamiseks:

- keelemudelite rakendamine piiratud mahuga treeningandmestiku tingimustes, et uurida, kas sisukokkuvõtteid on võimalik koostada ka olemasoleva vähese märgendatud andmestiku korral;
- EstSumi süsteemi laiendamine täiendavate lingvistiliste moodulitega, mille eesmärk on tõsta kokkuvõtete täpsust ja asjakohasust. Inspiratsiooni võeti nende arendamisel Joshi jt [8] poolt esitatud arhitektuurist, mille üheks oluliseks komponendiks oli semantilise analüüsi moodul.

Valminud programm on kättesaadav GitHubist [1].

Mudelite täpsuse (ingl *accuracy*) hindamiseks kasutati kattuvusprotsenti, mida kasutati ka Janar Saksa [4] töös EstSumi arendamisel. Joonisel 2 on toodud näide kattuvuse arvutamise algoritmist.



Joonis 2. Näide täpsuse hindamisest.

Rohelised kastid tähistavad lauseid, mis esinevad märgendatud kokkuvõttes, ning punased kastid viitavad laustele, mis kuuluvad küll algsesse artiklisse, kuid mitte märgendatud kokkuvõttesse.

[1] EstSum repositoorium: https://github.com/PMark-est/estsum

4.1 Siirdeõpe olemasoleva keelemudelitega

Siirdeõpe (ingl *transfer learning*) on tõhus meetod, mis võimaldab mudelite kiiremat kohandamist spetsiifilistele ülesannetele, kasutades eelnevalt treenitud keelemudeleid. Selle abil saab vähendada uue mudeli treenimisega seotud arvutuslikke kulusid ning parandada tulemuste täpsust, rakendades varasemalt õpitud teadmisi uutes kontekstides. Siirdeõppel on neli erinevat vormi [10]:

- Peenhäälestus (ingl *fine-tuning*) – selle meetodi puhul uuendatakse mudeli kaale, kohandades seda konkreetse ülesande jaoks. Peenhäälestamine nõuab suurt andmehulka ja on arvutuslikult nõudlik protsess, kuna kõik või osa mudeli kihte peavad läbima uuesti treenimise. Selle tulemusel võib mudel saavutada ülesandepõhise optimaalse jõudluse, kuid samas on see meetod ressursimahukas ja vähem paindlik, kui töötada erinevate ülesannetega.
- *Few-shot learning* – selle lähenemisega esitatakse mudelile vaid mõned näited konkreetsest ülesandest. Seda rakendatakse kasutades viipasid (ingl *prompts*). Viip on käsk, mis kasutaja annab keelemudelile. Andes viipa ülesandest koos vastavate vastustega, keelemudel õpib ülesande loogikat mõista ilma, et oleks vaja kaale uuendada.
- *One-shot learning* – mudelile esitatakse ainult üks näide. See võib sobida olukordadesse, kus ongi ainult üks andmepunkt ja mudel peab ülesande konteksti mõistma ühe viipa põhjal.
- *Zero-shot learning* – selle meetodiga ei anta mudelile ühtegi näidet ette, vaid viip sisaldab ainult ülesande püstituse kirjeldust.

Piiratud Þarvutusressursside ja vähese andmestiku tõttu ei ole peenhäälestamine võimalik, mistõttu on selle asemel kasutatud *few-shot* meetodit. Brown jt [10] märgivad, et selle peamine eelis peenhäälestamise ees (kehtib ka *one-shot* ja *zero-shot* puhul) on see, et mudeli kaale ei pea uuendama, mis vähendab märkimisväärselt arvutuslikku koormust ja võimaldab mudelit kiiremini rakendada erinevatele ülesannetele. Lisaks see sobib väga hästi väikeste andmekoguste puhul [11].

Katsetamisel kasutati Groqi pakutavat tasuta API-t, et testida keelemudeli toimivust. Need mudelid ei ole spetsiifiliselt treenitud eesti keele andmetel, mistõttu on oluline meeles pidada, et see võib tulemusi mõjutada. Lisades I ja II on näha 17.03.2025 seisuga nende poolt pakutavad keelemudeleid.

Kontekstiaken valitavatel mudelil peab olema piisavalt suur, et mahutada EstSum arenduskorpus, mis koosneb 16 246 sõnast ja kirjavahemärgist (arvutatud bash-skriptiga, vt lisa III), mis tähendab, et keskmiselt on artikli pikkus 812,3. OpenAI [12] andmetel on ingliskeelses tekstis ühe sõne (ingl *token*) keskmine pikkus umbes 0,75 sõna, kuid eesti keele puhul võib see suhe erineda. Samuti erinevad keelemudelid selle poolest, kuidas sõnestamist (ingl *tokenization*) rakendatakse, mis võib põhjustada sama teksti puhul erinevat sõnede arvu. Antud teenuse puhul mõjutab lisaks kontekstiakna suurusele ka mudelitele seatud päringulimiidid. Kõike seda arvesse võttes sobivad mudelid, mille kontekstiaken on vähemalt 32 000 (32K) sõne pikk. Mudelid, mille kontekstiaken on 16 000 sõne (16K) ei pruugi sobida, sest mällu ei mahuks kasutaja viip koos artikliga.

Tasuta versioonis on piirang, mis lubab päringut teha vaid piiratud arvu sõnedega (vt lisa IV), mis omakorda seab piirangud sisukokkuvõtete genereerimisele, kuna saab kasutada vähem andmeid treenimiseks. Katsetuste käigus selgus, et *few-shot* meetodit on võimalik rakendada vaid kolme artikli põhjal.

Siirdeõppe jaoks otsustati kasutada keelemudelit *llama-3.3-70b-versatile*, mille sõne piirang on 6000 sõne minutis. See tähendab, et minuti jooksul on võimalik viibata 6000 sõne ulatuses. See keelemudel sai valitud kolmel põhjusel:

1. see on neil *production* mudel, mis tähendab, et kui EstSumi tulevikus edasi arendatakse sama keelemudelit kasutades on suurem tõenäosus, et nad endiselt pakuvad seda;
2. *production* mudelite seast on see suurima kontekstiaknaga, milleks on 128 000 sõne (128K);
3. see on suure parameetrite arvuga ja suhteliselt uus mudel.

Mudeli õpetamiseks pakub Groqi API võimalust määrata viipade ajalugu. Viimane viip ajaloos on mõeldud kasutaja päringu jaoks, et mudel saaks sellele vastata. Lisaks on võimalik keelemudelile määrata rolli, mis suunab seda vastama nii nagu kasutaja soovib.

Mudeli ümberõpetamiseks kasutati lihtsat viipa, kus kasutaja päringuks oli tekst, millest sooviti kokkuvõtet, ning mudeli vastus oli vastava teksti kokkuvõte määratud stiilis. Õpetamiseks kasutati EstSum arenduskorpusest pärinevaid artikleid uudis1_alg, uudis2_alg ja uudis3_alg, millele vastasid käsitsi loodud kokkuvõtted uudis1, uudis2 ja uudis3. Testimise käigus pandi viipa ajaloo viimaseks sõnumiks artikkel, millest sooviti kokkuvõte. Katsetamisel kasutati testkorpusest järgmisi artikleid: uudis1_alg – uudis5_alg.

4.2 EstSum edasiarendus

Algati Janar Saksa [4] loodud EstSumi Pythoni versiooni ümberkirjutamisest. Uuendatud versioon valmis samuti Pythonis, kuid selle arendamisel rakendati objektorienteeritud programmeerimise võtteid ning järgiti häid tarkvaraarenduse tavasid [2]. Näiteks kasutati muutujatel tähenduslikke nimesid, mis kirjeldasid nende otstarvet, ning struktureeriti koodi nii, et iga meetod vastutab ühe kindla ülesande eest, jagades keerukamad funktsioonid väiksemateks abifunktsioonideks, mis parandavad loetavust ja hooldatavust. Selline lähenemine muudab tarkvara kergemini laiendatavaks ja hallatavaks, võimaldades tulevastel arendajatel seda lihtsamini mõista ja edasi arendada. Siiski tuleb märkida, et kuigi mõlemas versioonis kasutati sisukokkuvõtja loogika rakendamisel sama lähenemist, erines nende tulemuslikkus. Katsetamisel sama andmetega erinesid Saksa [4] tulemused uue rakenduse tulemustega; kasutati tabelis 1 olevaid parameetreid.

Tabel 1. Katsetamisel kasutatud parameetrid.

α	β	γ
0.4	0.4	0.2

Nende parameetrite väärtustega saavutas algne rakendus ligikaudu 62% täpsuse, samas kui uuendatud versiooni täpsus jäi ligikaudu 41% tasemele. See erinevus on suure tõenäosusega tingitud sellest, et uuendatud EstSumi kirjutamisel jäi mõningaid olulisi detaile kahe silma vahele. Varasem versioon EstSumist on autori hinnangul koostatud suhteliselt kaootiliselt, mis raskendas koodi mõistmist. Mõned peamised takistused olid näiteks:

- lühendid, mille tähendused ei olnud esmapilgul arusaadavad;
- globaalsed muutujad ja lipukesed, mis vähendasid koodi selgust ja loogilist ülesehitust;
- põhjendamata ja juhuslikud konstantide väärtused.

[2]Tavad tulevad isiklikust kogemusest, kuid neid toetavad mainekad isikud/asutused, nagu näiteks MIT ülikool: https://mitcommlab.mit.edu/broad/commkit/coding-mindset/#Convention

Uue versiooni lähtekood on kättesaadav autori GitHubi repositooriumis[3] ning varasem, Janar Saksa [4] loodud versioon, on avalikult saadaval tema GitHubi repositooriumis[4]. Edaspidi käsitletakse, kuidas uut versiooni sisukokkuvõtjast edasi arendati.

4.2.1 Semantiline analüüs Eesti Wordnetiga

Tartu Ülikooli arvutilingvistika uurimisrühma andmetel on wordnet struktuur, mis koosneb ühe- ja mitmesõnalistest sünonüümidest, koondades sõnu, mis väljendavad sama ideed. Neid tähenduslikult seotud sõnu organiseeritakse niinimetatud sünohulkadeks (sünonüümikomplektideks), mille kaudu on võimalik kirjeldada keele põhimõttelist ülesehitust. 2021. aasta oktoobri seisuga sisaldas Eesti wordnet ligikaudu 91 700 sünohulka, mida ühendas omavahel 306 000 semantilist suhet [13].

Antud lingvistilises moodulis rakendati semantilist analüüsi lihtsustatud kujul, hinnates sarnasust sõna tasandil pealkirja ja vastava lause vahel. Olgu pealkiri P, mis koosneb sõnadest $p_1, p_2, \ldots, p_n$, ja suvaline lause s, mis koosneb sõnadest $w_1, w_2, \ldots, w_m$. Iga lause kohta arvutati semantiline skoor S valemi 2 alusel. Lähtuti eeldusest, et laused, mille sõnad on sarnased või mõisteliselt seotud pealkirjas esinevate sõnadega, on suurema tõenäosusega artikli sisuga tihedamalt seotud ning seega olulisemad. Kui mõni sõna puudus wordneti andmestikust, määrati vastavaks väärtuseks 0. Kui sõna kohta leiti mitu sünonüümihulka, kasutati neist ainult esimest.

$$S(s) = \sum_{i=1}^{n} \frac{\sum_{j=1}^{m} \text{lch\_similarity}(p_i, w_j)}{m} \tag{2}$$

Leacock–Chodorow'i sarnasusmõõdik (ingl *LCH similarity*) võimaldab hinnata kahe sõna ideelist lähedust, tuginedes nendevahelisele lühimale teele taksonoomilises struktuuris, arvestades ka taksonoomia üldist sügavust [14]. Meetod põhineb eeldusel, et semantiliselt sarnased sõnad paiknevad wordneti hierarhias üksteisele lähedal. Arvutustes kasutati sõnade lemmakujusid, kuna wordnet ei tunneks muidu sõnu ära. See põhjustaks sõnale valesti hinnangu 0 määramist. Semantilist kaalu kasutati täiendava tegurina lause lõppkaalu arvutamisel, täiustades valemit 1 tähenduslähedu se arvesse võtmiseks. Lõpliku kaalu arvutamine toimub valemi 3 põhjal.

[3]Uus versioon EstSumist: https://github.com/PMark-est/estsum/blob/master/backend/estsum.py
[4]Vana versioon EstSumist: https://github.com/janarsaks/EstSum_development/blob/master/estsum2.py

$$W(s) = \alpha P(s) + \beta F(s) + \gamma K(s) + \delta S(s) \quad (3)$$

Lõpliku kaalu $W(s)$ arvutamiseks katsetati tabelis 2 olevate parameetritega.

Tabel 2. Parameetrid wordneti mooduli katsetamisel.

α	β	γ	δ
0.35	0.35	0.2	0.1

Esialgsete katsete põhjal järeldati, et erinevate parameetrite kombinatsioonide testimine ei ole mõistlik, kuna see aeglustas süsteemi märgatavalt. Sellest lähtuvalt otsustati moodul eemaldada.

4.2.2 Semantiline analüüs keelemudeliga

Keelemudelid on paindlikud ja neid saab rakendada mitmel viisil. Selles peatükis tutvustatakse, kuidas kasutati sama keelemudelit, kui peatükis 4.1, et arvutada lause seotust artikliga pealkirja põhjal. Samuti toetudes sellele ideele, et mida rohkem on lause seotud teemaga/pealkirjaga, seda olulisem see on.

Viipate koostamisel kasutati pakkimist (ingl *batching*), et mudelit efektiivsemalt ära kasutada. Tabel 3 kirjeldab, kuidas pakkimine välja paistab. Lause s lõppkaalu arvutamiseks kasutati valemit 4.

$$W(s) = S(s) \cdot (\alpha P(s) + \beta F(s) + \gamma K(s)), \quad (4)$$

kus $S(s)$ on keelemudeli poolt tagastatud väärtus vahemikus 0,0 ja 1,0. Sisukokkuvõtjat EstSum katsetati selle lisa mooduliga arenduskorpuses olevate artiklitega arvamus1 - arvamus10 ja uudis1-uudis10. Tulemustests arutatakse peatükis 5.

4.2.3 Veebilehe aadress sisendina

Veebis on artiklid kättesaadavad HTML-kujul. See tähendab, et iga artikli käsitsi ümbervormindamine sobivasse sisendvormingusse, mida EstSum suudab töödelda, oleks ajamahukas ja ebaefektiivne. Selle probleemi lahendamiseks loodi eraldi programm, mille eesmärk on veebilehelt vajaliku tekstilise sisu automaatne eraldamine ja selle teisendamine vormingusse, mis on EstSumi jaoks sobiv.

Tabel 3. Pakkimisega viip.

Role: Te olete sisukokkuvõtja ning teile on antud nimekiri lausetest. Võrdle iga antud lause selle lausega: "Mingi pealkiri". Anna hinnang 0 ja 1 vahel, kui seotud on laused.
User: 1. Lause 1 2. Lause 2 3. Lause 3 ... n. Lause n
System: 1. 0.1 2. 0.7 3. 0.4 ... n. 0

EstSumi kõrvale loodud abiprogramm võtab argumentidena veebilehe aadressi ning väljundfaili nime. Programm loeb määratud veebilehe sisu ning teisendab selle sobivasse SGML-vormingusse (vt lisa V), mida EstSum suudab töödelda sisukokkuvõtte loomiseks.

Implementatsioon on jäetud lihtsaks, arvestades asjaolu, et veebilehtedel ei ole ühtset HTML-märgistuse struktuuri ning nende kujundust uuendatakse aja jooksul sageli. HTML-i muutuv struktuur muudab täpse ja igal lehel toimiva andmeväljavõtte keeruliseks. Seetõttu baseerub rakendus peamiselt standardsetel HTML-elementidel. Seda tehakse kasutades teeki *BeautifulSoup*, mis võimaldab htmli lihtsat töötlemist. Esmalt kontrollitakse, kas veebileht sisaldab <main> või <article> märgendit, mille kaudu loodetakse leida artikli põhisisu. Artikli pealkiri eraldatakse <h1> märgendist ning lõigud <p> elementide abil.

Lausete eraldamiseks lõikudest kasutatakse teeki estnltk. Iga leitud lause paigutatakse väljundfailis <s> ja </s> märgendite vahele, lõigud aga ja vahele. Hetkel puudub toetus alampealkirjade või muude keerukamate struktuurielementide käsitlemiseks, kuid selle lisamine võiks olla edasine arendussuund.

5. Tulemused

Käesolevas peatükis esitatakse tulemused, mis saadi EstSumi täiustatud versiooni katsetamisel – nii lisatud lingvistiliste moodulite kui ka keelemudeli põhise lähenemise kaudu. Lisaks käsitletakse võimalikke tulevikusuundi, mille kaudu võiks süsteemi edasi arendada. Varasemast EstSumi arendusest on olemas arenduskorpus, mis koosneb kahekümnest artiklist – kümnest arvamusartiklist ja kümnest uudisartiklist; ning testkorpus, mis koosneb kümnest artiklist - viiest arvamusartiklist ja viiest uudisartiklist.

5.1 Lingvistiliste moodulite tulemused

Lingvistiliste moodulite katsetamisel kasutati arenduskorpusest kõiki artikleid. Esimene testitud moodul oli wordnetil põhinev semantiline analüüs. Selle lisamine EstSumi arhitektuuri tõstis süsteemi keskmist täpsust ligikaudu ühe protsendipunkti võrra (vt tabel 4). Kuigi tulemuses ilmnes paranemine, ei ole see arvestades tekkivaid jõudlusprobleeme piisav, et mooduli kasutamist tulevikus põhjendada. Wordneti rakendamine aeglustas süsteemi märgatavalt – ühe sisukokkuvõtte genereerimine võttis aega 30 sekundist kuni 2 minutini, keskmiselt ligikaudu üks minut. Ilma on keskmiselt 0,5 sekundi kanti.

Lisaks selgus, et enamiku sõnapaaride puhul tagastas wordnet sarnasuse väärtuseks null, kuna sõnad ei olnud omavahel seotud või üks neist puudub wordnetist täielikult. Samuti esines juhtumeid, kus mõne sõna jaoks tagastati mitu erinevat sünohulka, millest valiti ainult esimene. On võimalik, et parema tulemuse saavutamiseks saaks katsetada ka alternatiivide põhjal, kuid see muudaks arvutuse veelgi aeglasemaks ja ei oleks praktikas otstarbekas.

Teine moodul, mis põhines semantilisel sisendil keelemudeli abil, parandas keskmiselt tulemuslikkust ligikaudu 4 protsendipunkti võrra (vt tabel 4). Kuigi see ei ole hüppeline kasv, on märkimisväärne, et see moodul ei mõjutanud süsteemi kiirust oluliselt – keskmine töötlusaeg oli umbes 1,5 sekundit artikli kohta. Kuna tegemist on keelemudeli põhise lähenemisega, on võimalik selle täpsust veelgi parandada viipade kujundamine (ingl *prompt engineering*) abil.

Keelemudelil põhinev moodul on tehniliselt tasakaalustatud lahendus, mis ühendab väikese täpsuse kasvu ning vähese arvutuskoormuse. Kuigi ligikaudu 4-protsendiline paranemine ei ole väga suur, on märkimisväärne, et see saavutati ilma süsteemi töökiirust oluliselt mõjutamata. See näitab, et semantilise tähenduse hindamine keelemudeli kaudu suudab pakkuda sisulist lisaväärtust ka ilma olemasolevat arhitektuuri põhjalikult ümber kujundamata. Samas kaasnevad keelemudelite kasutamisega teatud piirangud – eriti probleem hallutsinatsioonidega (ingl

hallucinations), mille puhul mudel võib esitada väljamõeldud või vigadega teavet. Samuti esineb varieeruvust genereeritud vastustes isegi identse viipa korral, mis tähendab, et semantilised kaalud ei pruugi olla korduvkasutatavad. Kuigi vastuse stabiilsust on võimalik mõnevõrra parandada viipade täpsustamise kaudu, jääb faktiks, et sellise ülesande puhul puudub üks kindel tõeväärtuslik väljund, mistõttu on iga päringuga tulemused paratamatult varieeruvad.

Mõlema mooduli puhul jääb avatuks ka küsimus, kuidas saaks täiustada EstSumi arhitektuuri selliselt, et semantilist teavet saaks süsteemi sisemises loogikas paremini ära kasutada.

5.2 Keelemudeli tulemused

Keelemudeli eraldiseisval kasutamisel (ilma EstSumi reeglitel põhineva loogikata) saavutati keskmiselt veidi madalam täpsus – umbes 0,5 protsendipunkti võrra väiksem, kui EstSumi põhiversioonil (vt tabel 5). Nagu ka semantilise mooduli puhul, sõltuvad need tulemused suuresti kasutatud viipadest. Käesolevas töös kasutati *few-shot* lähenemist, kus mudelile anti kolm näidet arenduskorpusest - arvamus1, arvamus2 ja arvamus3. Selline piirang tulenes Groqi teenuse tehnilistest piirangutest (vt lisa IV).

Hoolimata piirangutest näitavad tulemused, et isegi väga vähese treeningandmestikuga on võimalik saavutada konkurentsivõimeline tulemus. See avab mitmeid edasiarenduse võimalusi:

- Eesti keelele kohandatud keelemudelite kasutamine, näiteks Tartu Ülikooli loodud Llammas, mis on peenhäälestatud Llama2-7b [15].
- Keelemudeli iseseisev peenhäälestamine eesti keele andmetel, näiteks kasutades eesti keele koondkorpust.
- Rikkalikum *few-shot* õpe, kasutades rohkem näiteid ja paremini vormindatud viipasid, et mudel suudaks paremini üldistada ülesande loogikat.

Kattuvust oleks võimalik parandada ka nii, et semantilist analüüsi rakendatakse lisaks reeglipärasele analüüsile – kas pärast seda, kui suurema kaaluga laused on juba välja valitud, või vastupidi: esmalt valib laused keelemudel ning reeglipõhine analüüs rakendatakse ainult nendele.

5.3 Tulevikusuund

EstSumi täiustamisel on mitmeid suundi, mis väärivad edasist uurimist ja arendust. Eriti keelemudelil põhinev semantiline analüüs, mis parandas kokkuvõtete täpsust ilma süsteemi töökiirust märgatavalt mõjutamata. Kuna tegemist on keelemudeli abil arvutatava semantilise

Tabel 4. Lingvistiliste moodulite tulemused arenduskorpuse artiklitel.

Artikkel arenduskorpusest	**Baas**	**Lingvistiline moodul wordnetiga**	**Lingvistiline moodul keelemudeliga**
Uudis 1	33,33%	30,32%	40,00%
Uudis 2	63,64%	67,50%	72,73%
Uudis 3	72,73%	72,20%	77,78%
Uudis 4	30,00%	53,95%	37,5%
Uudis 5	16,67%	40,48%	36,36%
Uudis 6	62,5%	48,50%	42,86%
Uudis 7	57,14%	52,84%	85,71%
Uudis 8	40,00%	28,40%	40,00%
Uudis 9	20,00%	50,34%	22,22%
Uudis 10	50,00%	54,11%	71,43%
Arvamus 1	15,52%	24,47%	25,00%
Arvamus 2	34,04%	32,39%	44,44%
Arvamus 3	63,70%	39,00%	69,23%
Arvamus 4	63,59%	42,22%	75,00%
Arvamus 5	35,34%	30,25%	36,36%
Arvamus 6	24,4%	19,53%	23,81%
Arvamus 7	34,9%	27,63%	18,18%
Arvamus 8	25,89%	30,84%	27,27%
Arvamus 9	27,55%	46,37%	42,86%
Arvamus 10	56,27%	52,31%	25,00%
Keskmine	**41,36%**	**42,19%**	**45,69%**

Tabel 5. Keelemudeli tulemused.

Artikkel arenduskorpusest	**Baas**	**Keelemudel**
Uudis 1	33,36%	50,00%
Uudis 2	66,67%	60,00%
Uudis 3	42,86%	71,43%
Uudis 4	71,73%	41,18%
Uudis 5	50,0%	41,66%
Keskmine	**53,46%**	**52,84%**

analüüsiga, siis on selle tulemusi võimalik veelgi parandada viipade kujundamise kaudu. Edaspidi võiks uurida, kuidas integreerida seda tüüpi semantilist teavet sügavamalt EstSumi sisemisse arhitektuuri, et saaks paremini arvestada lause tähendust ja seost ülejäänud artikliga.

Keelemudelite iseseisev kasutamine näitas, et isegi piiratud treeningandmestikuga on võimalik saavutada tulemusi, mis on võrreldavad reeglitel põhineva EstSumi täpsusega. Tulevikus võiks kaaluda järgmisi suundi:

- eesti keelele kohandatud keelemudelite rakendamine, näiteks Tartu Ülikooli loodud Llammas, mis on peenhäälestatud versioon Llama2-7b mudelist;
- olemasolevate keelemudelite sihipärane peenhäälestamine eesti keele andmestikul, kasutades näiteks eesti keele koondkorpust;
- *few-shot* õppe edasiarendus: rohkemate näidete kasutamine ja viipade kujundamine.

Lisaks on oluline analüüsida, miks jäi uuendatud versiooni baastäpsus märgatavalt madalamaks võrreldes varasema EstSumi versiooniga. Võimalike tehniliste puudujääkide tuvastamine ja kõrvaldamine võimaldaks tõsta süsteemi üldist tulemuslikkust ligikaudu 20% võrra.

6. Liidese arendus

Selleks, et muuta EstSum kättesaadavaks ka kasutajatele, kellel puuduvad tehnilised oskused või võimalused programmi käsitsi käivitada, loodi töö raames veebirakendus, mis võimaldab sisukokkuvõtteid genereerida lihtsalt ja kasutajasõbralikult. Kuna lahendus on veebipõhine, ei ole vaja tarkvara eraldi paigaldada ning teenusele pääseb ligi erinevates seadmetes ja asukohtades. Kokkuvõtte loomine toimub artikli veebiaadressi sisestamise teel, ilma et kasutaja peaks kasutama käsurida või tegelema tehnilise seadistamisega. Lisaks sisestusväljale saab kasutaja määrata soovitud kokkuvõtte pikkuse, arendajatele on aga loodud täiendav funktsionaalsus lausekaalude muutmiseks, et kohandada kokkuvõtete koostamisel kasutatud kaalusid.

Veebirakenduse kujundus loodi Figma prototüüpimisplatvormil, mis võimaldab kiiret iteratiivset arendust. Selle lõplikku kujundust on näha lisas VI. Rakenduse tehniline ülesehitus kasutab raamistikku Django 5.2 serveripoolseks töötluseks ning React 19.1.0 raamistikku kasutajaliidese loomiseks. Kooskõlas *React* raamistikuga kasutati teeke Material UI ja Tailwind CSS Need tehnoloogiad valiti välja sellepärast, et autoril on nendega varasem kogemus.

Django on programmeerimiskeelel Python põhinev raamistik, mis aitab arendajatel luua veebirakenduste serveripoolseid (ingl *back-end*) komponente. See tähendab, et Django hoolitseb selle eest, kuidas andmed liiguvad veebibrauseri ja serveri vahel.

React on programmeerimiskeele JavaScript teek, mida kasutatakse kasutajaliideste (ingl *front-end*) loomiseks. Selle abil on võimalik arendada dünaamilisi ja interaktiivseid veebirakendusi, mis võimaldavad kasutajal süsteemiga reaalajas suhelda. Arendusprotsessi tõhustamiseks ja kasutajakogemuse parandamiseks kasutatakse sageli täiendavaid tööriistu, nagu Material UI ja Tailwind CSS.

Material UI on kasutajaliidese komponentide kogu, mis pakub eeldefineeritud ja visuaalselt esteetilisi elemente, lihtsustades seeläbi arendust ja tagades disainijärjepidevuse. Tailwind CSS võimaldab aga kujunduselementide kirjeldamist otse HTML-koodis klassinimede kaudu, ilma vajaduseta eraldiseisvate CSS-failide järele, mis muudab stiilide rakendamise paindlikumaks ja kiiremaks.

Reacti arhitektuur põhineb komponentidel, mis võimaldavad rakenduse jagamist väiksemateks, iseseisvalt hallatavateks ja taaskasutatavateks osadeks. Iga komponent võib täita spetsiifilist funktsiooni, näiteks vastutada sisestusväljade, nuppude või tulemuste kuvamise eest.

7. Kokkuvõte

Käesoleva töö eesmärk oli katsetada eestikeelse sisukokkuvõtja EstSum tulemuslikkuse parandamist semantilise analüüsi abil. Lisaks moodulite lisamisele uuriti ka keelemudelite kasutatavust sisukokkuvõtete loomiseks. Eesmärgiks seati ka veebirakenduse arendamine, mis võimaldab süsteemi kasutada mittetehnilistel kasutajatel.

Töös katsetati kahte uut lingvistilist moodulit: wordnetil põhinevat ja keelemudelil põhinevat semantilist analüüsi. Nende mõju hinnati arenduskorpuse abil. Wordneti rakendamine parandas tulemusi vähe, umbes protsendi võrra, ning tõi kaasa märgatava jõudluslanguse. Keelemudelil põhinev moodul parandas täpsust umbes nelja protsendipunkti võrra ning säilitas süsteemi töökiiruse. Lisaks katsetati keelemudelite kasutamist sisukokkuvõtete loomiseks *few-shot* meetodil, mille tulemusel saavutati sarnane täpsus EstSumi baasversiooniga.

Töö tulemusel loodi ka objektorienteeritud arhitektuuril põhinev laiendatav versioon EstSumist, mille edasiarendus on lihtsustatud ja tulevikukindel. Edasiste arenduste puhul on soovitatav keskenduda viipade täpsustamisele, suuremate või eesti keelele kohandatud keelemudelite rakendamisele ning moodsamate arhitektuuride ja moodulite integreerimisele.

Viited

[1] Luhn H. P. The Automatic Creation of Literature Abstracts. *IBM Journal of Research and Development* (1958), lk 159–165.

[2] Edmundson H. P. New Methods in Automatic Extracting. *J. ACM* (1969), lk 264–285.

[3] Müürisep K. ja Mutso P. ESTSUM - Estonian Newspaper Texts Summarizer. *Proceedings of the Second Baltic Conference on Human Language Technologies* (2005), lk 311–316. https://www.researchgate.net/publication/250152518_ESTSUM_-_ESTONIAN_NEWSPAPER_TEXTS_SUMMARIZER (08.12.2024).

[4] Saks J. Eestikeelsete tekstide sisukokkuvõtja EstSum edasiarendamine. TÜ arvutiteaduse instituudi bakalaureusetöö. 2018. https://dspace.ut.ee/server/api/core/bitstreams/2076e963-ccea-4b41-b81d-693fa37048db/content (08.12.2024).

[5] Vaishali, Sehgal G. ja Dixit P. A Comprehensive Study of Automatic Text Summarization Techniques. *2024 International Conference on Emerging Innovations and Advanced Computing (INNOCOMP)* (25. mai 2024), lk 688–694. https://research-ebsco-com.ezproxy.utlib.ut.ee/linkprocessor/plink?id=07658682-a9c4-3302-9f58-3f6580625227 (08.12.2024).

[6] Parth Mehta P. M. From Extractive to Abstractive Summarization: A Journey. Springer Nature, 2019.

[7] Murel, J. What is Text Summarization? 2023. https://www.ibm.com/think/topics/text-summarization (09.05.2025).

[8] Joshi A., Fidalgo E., Alegre E. ja Alaiz-Rodriguez R. RankSum: An Unsupervised Extractive Text Summarization Based on Rank Fusion. *arXiv* (2024). https://research-ebsco-com.ezproxy.utlib.ut.ee/linkprocessor/plink?id=06c161c2-05e3-35f7-8263-2820b4db5c3f (08.12.2024).

[9] Gupta S. ja Gupta S. K. Abstractive summarization: An overview of the state of the art. *Expert Systems with Applications* 121 (2019), lk 49–65. https://www.sciencedirect.com/science/article/pii/S0957417418307735.

[10] Brown T. B., Mann B., Ryder N., Subbiah M., Kaplan J., Dhariwal P., Neelakantan A., Shyam P., Sastry G., Askell A., Agarwal S., Herbert-Voss A., Krueger G., Henighan T., Child R., Ramesh A., Ziegler D. M., Wu J., Winter C., Hesse C., Chen M., Sigler E., Litwin M., Gray S., Chess B., Clark J., Berner C., McCandlish S., Radford A., Sutskever I. ja Amodei D. Language Models are Few-Shot Learners. *arXiv* (2020). eprint: 2005.14165. https://arxiv.org/abs/2005.14165 (09.03.2025).

[11] Wang Y., Yao Q., Kwok J. T. ja Ni L. M. Generalizing from a Few Examples: A Survey on Few-Shot Learning. *ACM Computing Surveys* (2020), lk 1–34. https://dl.acm.org/doi/10.1145/3386252 (11.05.2025).

[12] OpenAI. What are tokens and how to count them. 2023. https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them (11.05.2025).

[13] University of Tartu. Eesti keele teksaurus. 2023. https://www.cl.ut.ee/ressursid/teksaurus/ (09.05.2024).

[14] Natural Language Toolkit Project. Sample usage for wordnet. 2023. https://www.nltk.org/howto/wordnet.html (09.05.2024).

[15] Kuulmets H.-A., Purason T., Luhtaru A. ja Fishel M. Teaching Llama a New Language Through Cross-Lingual Knowledge Transfer. *Findings of the Association for Computational Linguistics: NAACL 2024*. Association for Computational Linguistics, 2024, lk 3309–3325. https://aclanthology.org/2024.findings-naacl.210.

Lisad

Siia saate teha eraldi alamsektsioonid näiteks oma lõputöö sõnastiku, kaasapandud failide kirjelduse, kasutajajuhendi, suurte piltide ja tabelite jaoks.

I. Groq production keelemudelid - s.o mudelid, mida pidevalt hostivad

MODEL ID	DEVELOPER	CONTEXT WINDOW (TOKENS)	MAX COMPLETION TOKENS	MAX FILE SIZE	MODEL CARD LINK
distil-whisper-large-v3-en	HuggingFace	-	-	25 MB	Card
gemma2-9b-it	Google	8,192	-	-	Card
llama-3.3-70b-versatile	Meta	128K	32,768	-	Card
llama-3.1-8b-instant	Meta	128K	8,192	-	Card
llama-guard-3-8b	Meta	8,192	-	-	Card
llama3-70b-8192	Meta	8,192	-	-	Card
llama3-8b-8192	Meta	8,192	-	-	Card
mixtral-8x7b-32768	Mistral	32,768	-	-	Card
whisper-large-v3	OpenAI	-	-	25 MB	Card
whisper-large-v3-turbo	OpenAI	-	-	25 MB	Card

II. Groq preview keelemudelid - s.o ajutised mudelid, mille hostimise võivad lõpetada

qwen-qwq-32b	Alibaba Cloud	128K	-	-	Card
mistral-saba-24b	Mistral	32K	-	-	Card
qwen-2.5-coder-32b	Alibaba Cloud	128K	-	-	Card
qwen-2.5-32b	Alibaba Cloud	128K	-	-	Card
deepseek-r1-distill-qwen-32b	DeepSeek	128K	16,384	-	Card
deepseek-r1-distill-llama-70b-specdec	DeepSeek	128K	16,384	-	Card
deepseek-r1-distill-llama-70b	DeepSeek	128K	-	-	Card
llama-3.3-70b-specdec	Meta	8,192	-	-	Card
llama-3.2-1b-preview	Meta	128K	8,192	-	Card
llama-3.2-3b-preview	Meta	128K	8,192	-	Card
llama-3.2-11b-vision-preview	Meta	128K	8,192	-	Card
llama-3.2-90b-vision-preview	Meta	128K	8,192	-	Card

III. Bash skript tokenite arvu leidmiseks arenduskorpuses

```
#!/bin/bash

total_tokens=0

for file in ./arenduskorpus/uudised/uudis*/uudis*_alg.txt; do
      if [ -f "$file" ]; then
      token_count=$(grep -oE '\w+|[[:punct:]]' "$file" | wc -l)
      echo "$file: $token_count tokens"
      total_tokens=$((total_tokens + token_count))
      fi
done

for file in ./arenduskorpus/arvamused/arvamus*/arvamus*_alg.txt; do
      if [ -f "$file" ]; then
      token_count=$(grep -oE '\w+|[[:punct:]]' "$file" | wc -l)
      echo "$file: $token_count tokens"
      total_tokens=$((total_tokens + token_count))
      fi
done

echo "Total tokens in all _alg.txt files: $total_tokens"
```

IV. Tasuta mudeli piirangud. RPM on requests per minute; RPD on requests per day; TPM on tokens per minute; TPD on tokens per day

MODEL ID	RPM	RPD	TPM	TPD	ASH	ASD
deepseek-r1-distill-llama-70b	30	1,000	6,000	-	-	-
deepseek-r1-distill-qwen-32b	30	1,000	6,000	-	-	-
distil-whisper-large-v3-en	20	2,000	-	-	7,200	28,800
gemma2-9b-it	30	14,400	15,000	500,000	-	-
llama-3.1-8b-instant	30	14,400	6,000	500,000	-	-
llama-3.2-1b-preview	30	7,000	7,000	500,000	-	-
llama-3.2-3b-preview	30	7,000	7,000	500,000	-	-
llama-3.2-11b-vision-preview	30	7,000	7,000	500,000	-	-
llama-3.2-90b-vision-preview	15	3,500	7,000	250,000	-	-
llama-3.3-70b-specdec	30	1,000	6,000	100,000	-	-
llama-3.3-70b-versatile	30	1,000	6,000	100,000	-	-
llama-guard-3-8b	30	14,400	15,000	500,000	-	-
llama3-8b-8192	30	14,400	6,000	500,000	-	-
llama3-70b-8192	30	14,400	6,000	500,000	-	-
mistral-saba-24b	30	1,000	6,000	-	-	-
qwen-2.5-32b	30	1,000	6,000	-	-	-
qwen-2.5-coder-32b	30	1,000	6,000	-	-	-
qwen-qwq-32b	30	1,000	6,000	-	-	-
whisper-large-v3	20	2,000	-	-	7,200	28,800
whisper-large-v3-turbo	20	2,000	-	-	7,200	28,800

V. Märgendatud kokkuvõtte formaat

```
<div0><head><hi rend="bold">Paksus kirjas pealkiri</hi></head>
<p>
<s>Lõigud "<p>" ja "</p>" märgendite vahel</s>
<s>Laused "<s>" ja "</s>" märgendite vahel</s>
</p>
<div><head>Alampealkiri</head>
<p>
<s><hi rend="bold">Paksus kirjas lause.</hi></s>
<s><hi rend="italic">Kaldkirjas lause.</hi></s>
</p>
</div>
</div0>
```

VI. EstSumi liidese disain Figmas

VII. Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, **Marko Peedosk**,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose
   Eestikeele sisukokkuvõtja edasiarendamine,
   mille juhendaja(d) on **Sven Aller**,
   reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada Tartu Ülikooli digitaalarhiivi kuni autoriõiguse kehtivuse lõppemiseni;

2. annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi kaudu Creative Commonsi litsentsiga CC BY NC ND 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni;

3. olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile;

4. kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Marko Peedosk 14.05.2025