

Eksperimentinės plėtros veiklos ataskaita

Projekto pavadinimas: Nurodyti

Projekto numeris: J05-LVPA-K-01-XXXX

Projekto vykdytojas: *UAB „Omnisend“*

Projekto partneriai: UAB „Tokenmill“

Veiklos numeris: Nurodyti

Veiklos pavadinimas: *Nurodyti*

Ataskaitos data: 2019-06-14

Lapų skaičius: Nurodyti

Turinys

[1. Veiklos tikslas 3](#_Toc11242468)

[2. Veiklos užduotys 3](#_Toc11242469)

[3. Esamo NLG sprendimo architektūros apžvalga 3](#_Toc11242470)

[3.1. Architeltūros komponentai 3](#_Toc11242471)

[3.2. Technologiniai komponentai 5](#_Toc11242472)

[4. Testavimo strategijos parengimas 5](#_Toc11242473)

[4.1. Veiklos aprašas 5](#_Toc11242474)

[4.2. Veiklos vykdymo ega 7](#_Toc11242475)

[5. Abstrakti prasmės reprezentacija dokumento plane 7](#_Toc11242476)

[5.1. Veiklos aprašas 7](#_Toc11242477)

[5.2. Veiklos vykdymo eiga 8](#_Toc11242478)

[5.3. Rezultatai 11](#_Toc11242479)

[6. Vartotojo modelio komponentas 12](#_Toc11242480)

[6.1. Veiklos aprašas 12](#_Toc11242481)

[6.2. Veiklos vykdymo eiga 12](#_Toc11242482)

[6.3. Rezultatai 13](#_Toc11242483)

[7. Galutinio teksto generavimas 14](#_Toc11242484)

[7.1. Veiklos aprašas 14](#_Toc11242485)

[7.2. Veiklos vykdymo eiga 14](#_Toc11242486)

[7.3. Rezultatai 14](#_Toc11242487)

[8. Įvykdyti paslaugų pirkimai 15](#_Toc11242488)

[9. Užduočių atlikimas 15](#_Toc11242489)

[10. Pasiekti rezultatai 16](#_Toc11242490)

[11. Nepasiekti rezultatai 16](#_Toc11242491)

[12. Veiklos vykdymo metu atliktų tyrimų eigos pakeitimai 16](#_Toc11242492)

[13. Rezultatų naujumas ir nauda projektui 16](#_Toc11242493)

[Bibliografija 17](#_Toc11242494)

# Veiklos tikslas

Sukurti metodiką ir prototipą prekių, elektroninės prekybos sistemose, aprašymų generavimui, panaudojant natūralios kalbos generavimo (NLG) ir dirbtinio intelekto (AI) metodus.

# Veiklos užduotys

Šiame etape buvo apibrėžtos tokios veiklos užduotys:

* 1. *Testavimo strategijos parengimas* – teksto generavimo sistemų testavimas negali naudoti tų pačių metodų, kurie naudojami deterministinių programinės įrangos komponentų testavimui. NLG sistema turi būti testuojama naudojant:
     1. jos rezultato loginės formos analizę;
     2. teksto variantų teisingumo patikrinimą.
  2. *Abstrakti prasmės reprezentacija dokumento plane* – plano redaktorius veikia kaip aukšto lygio programavimo kalba, todėl norint užtikrinti jo veikimo nuoseklumą, turi būti naudojama formalus tokios programavimo kalbos apibrėžimas.
  3. *Vartotojo modelio komponentas* – generuojamas tekstas turi adaptuotis prie skirtingų vartotojų galimai skirtinos reakcijos į tekste pateikiamus faktus. Todėl NLG sistema turi leisti valdyti teksto struktūrą ir žodyną, priklausomai nuo pasirikto vartotojo modelio.
  4. Galutinio teksto generavimas (Angl: *Surface Realizer*) – ši veikla buvo atliekama pirmame projekto etape (2018-09-25 ataskaita [12], skyrius *4.3 Sakinių realizavimas*), šio etapo metu ji buvo peržvelgta atsižvelgiant į atnaujintus produkto reikalavimus.

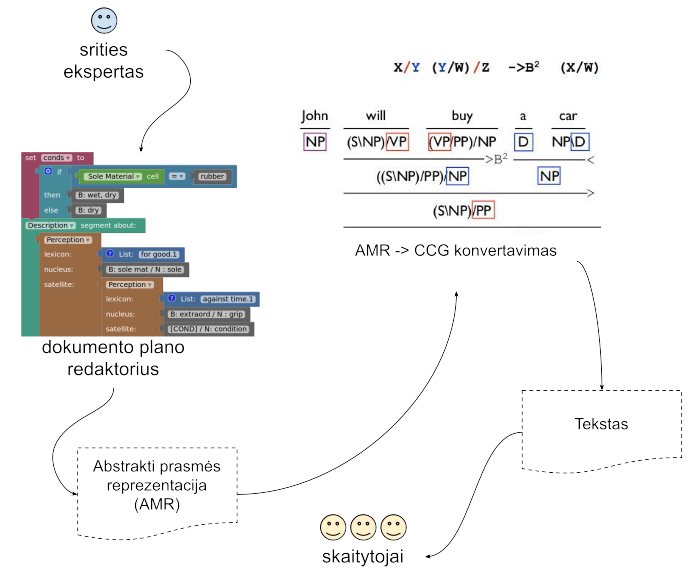
# Esamo NLG sprendimo architektūros apžvalga

Darbinė sprendimo versija yra pasiekiama šiuo adresu:

<http://accelerated-text-app.s3-website.eu-central-1.amazonaws.com/>

## Architektūros komponentai

Kuriamas teksto generavimo produktas šiuo metu veikia pagal žemiau pateiktą, supaprastintą sprendimo architektūrą.

  
Pav 1: NLG architektūros supaprastinta schema

Yra dvi NLG sistemos naudotojų grupės:

* 1. **Srities ekspertas** – Asmuo kuriantis pranešimo loginę struktūrą, *kas ir kodėl turi būti pasakyta*. Srities ekspertas nesirūpina teksto gramatine struktūra ir realizavimo detalėmis, jo tikslas yra specifikuoti kokie pranešimai turi būti perduoti galutiniam vartotojui.
  2. **Skaitytojai** – Įvairios skaitytojų grupės, kurioms, pagal jų specifiką yra kuriami tekstai, pagal srities eksperto aprašytą loginę pranešimo struktūrą.

*Srities ekspertas* naudoja *Plano redaktorių* (aprašytą šio projekto 2019 kovo 13 dienos ataskaitoje [1]). *Skaitytojai* tekstą gauna per NLG sistemos teikiamą programinę sąsają (API). Pats teksto rodymas yra NLG sistemą naudojančios komponentės atsakomybė. Mūsų atveju, tai dažniausiai yra el-parduotuvės tinklalapis.

Dokumento plano transformacija į galutinį tekstą atliekama naudojant tris NLG komponentus:

* 1. **Abstrakti prasmės reprezentacija** *(Angl.: Abstract Meaning Representation – AMR).* Negramatinė teksto prasmės reprezentacija. Detaliau šio dokumento 5-ame skyriuje.
  2. **Kombinatyvi kategorinė gramatika** *(Angl.: Combinatory Categorial Grammar – CCG).* Gramatika, aprašanti, pranešimo realizavimą. Plačiau CCG buvo aprašytas šio projekto 2019 kovo 13 dienos ataskaitoje [1], penktame dokumento skyriuje).
  3. **Baigtinio sakinio realizavimas** *(Angl.: Surface Realization)*. Nors labai svarbi NLG proceso dalis, šio projekto rėmuose mes nekreipiam daug dėmesio stilistikos, sakinio leksinės struktūros ir panašių klausimų sprendimui. Jie labiau aktualūs generuojant ilgesnius ir ne duomenis aprašančius tekstus. Pvz. teksto santraukos, žurnalistiniai straipsniai. Plačiau apie tai *7 Galutinio teksto generavimas* skyriuje.

Visą šiame projekte kuriamą NLG sprendimą būtų galima aprašyti, kaip turintį vienintelį tikslą sukurti priemones kurios leidžia automatizuoti AMR konvertavimą į CCG. Plačiau apie tai šio dokumento *5 Abstrakti prasmės reprezentacija dokumento plane* skyriuje.

# Technologiniai komponentai

Projekto vystymo metu buvo bandomi keli duomenų tvarkymo ir teikimo kliento sistemoms būdai. Nors šio dokumento tikslas nėra aprašyti žemesnio lygio techninius sprendimus, du technologiniai pasirinkimai verti paminėjimo, nes jie atspindi NLG sprendimo poreikius.

1. **Duomenų saugykla**. Šio projekto 2018-12-17 ataskaitoje [2] buvo aprašoma RAGS architektūra *(Angl: Reference Architecture for Generation Systems)*, kurios pagrindu mes kuriame savo sprendimą. Pažymėtina, kad naudojamos duomenų struktūros yra:
   * 1. grafo duomenų tipo;
     2. duomenys atspindi lingvistinę specifiką, kuri reikalauja labai lanksčių duomenų reprezentavimo struktūrų.

Todėl, mums reikėjo ieškoti duomenų saugyklos technologijos leidžiančios realizuoti šiuos reikalavimus. Papildomas, trečiasis reikalavimas buvo technologijos pereinamumas per mūsų naudojamą *Clojure* [3] programavimo kalbos aplinką. Visus šiuos reikalavimus puikiai atitiko *Datomic* [4]duomenų bazė.

1. **API**. Kaip ir duomenų struktūros atveju, sudėtingi lingvistiniai duomenys ir užklausų procesai reikalauja technologijos, leidžiančios lanksčiai aprašyti duomenų užklausas. Nors 2018-12-17 ataskaitoje [2] skyrius – 6, mes buvome pasirinkę tradicinę *RESTfull* [5] sąsaja, projekto pilnam realizavimui perėjome prie *GraphQL* [6] architektūra paremto API.

# Testavimo strategijos parengimas

## Veiklos aprašas

Teksto generavimo sistemos yra sunkiai testuojamos. Kaip pasakyti ar sugeneruotas tekstas yra prasmingas, ar viena gramatinė klaida reiškia, kad visas tekstas yra neteisingas? Jei generuojami tekstai yra akivaizdžiai generuoti mašinos, ar tai yra blogai?

Visi atsakymai į šiuos klausimus yra subjektyvūs ir itin priklausomi nuo kiekvieno panaudojimo atvejo. *Google Translate* vertimai yra dažnai, tiek gramatiškai tiek prasmės teisingumo perdavimo prasme, nepilnai teisingi. Tačiau tai visiškai mus tenkina verčiant naujienų straipsnius. Pardavimo sutarties vertimui, greičiausiai, *Google Translate* nenaudotime.

NLG sistemos paprastai testuojamos naudojant mašininiam vertimui sukurtas testavimo metrikas. *BLEU* [9] yra viena iš labiausiai paplitusių metrikų. Ji remiasi pažodiniu visų sakinio žodžių lyginimu. Imamas sugeneruotas sakinys ir N žmogaus parašytų sakinių. Kiekvienoje sakinio pozicijoje yra tikrinama, kiek yra sutampančių žodžių, sutapimo proporcija ir yra BLEU įvertis.

Šiame projekte mes atsisakome pilnai automatinio generuoto teksto vertinimo. Pagal mūsų kuriamo produkto panaudojimo specifiką mums itin svarbi, tikslios prasmės perdavimo, teksto savybė. Šios teksto savybės BLEU ir kitos panašios metrikos neužtikrina. Taip pat, remiantis Chaganty A. T. (2018) [10] automatinio testavimo analize, tokio testavimo kaštai yra tokie patys ar didesni, kaip pilnai rankinio testavimo išlaidos.

Testavimo neautomatizavimas nereiškia, kad testavimas nedaromas. Kuriamos NLG sistemos testavimas pradedamas atlikti naudojantis *Abstract Meaning Representation* [7] specifikacijoje naudojamomis, prasmės reprezentavimo ir jos susiejimo su sakiniu, konstrukcijomis (plačiau apie AMR panaudojimą – šio dokumento *5* *Abstrakti prasmės reprezentacija dokumento plane* skyriuje).

Testavimo metodologija, leidžianti objektyviai vertinti esamą teksto generavimo kokybę, tampa pagrindiniu šio projekto tikslu. Neturėdami pamatuojamu generavimo įverčių mes negalėsime matuoti kuriamos sistemos progreso.

## Veiklos vykdymo ega

AMR specifikacija [11] eilei pavyzdinių sakinių aprašo tokias, teksto ir loginės struktūros tapatinimo, konstrukcijas:

|  |  |  |
| --- | --- | --- |
| **#** | **Sakiniai** | **AMR** |
| 1 | *The boy saw the girl who wanted him* | (s / see-01  :ARG0 (b / boy)  :ARG1 (g / girl  :ARG0-of (w / want-01  :ARG1 b))) |
| 2 | *The boy saw the girl who he was wanted by* |
| 3 | *The girl who wanted the boy was seen by him* |

Testavimo procesas susideda iš dviejų dalių:

* + *Pirmą*, tikrinama kiek sakinių NLG sistema gali realizuoti. Imant aukščiau pateiktą pavyzdį, tarkim, teisingai (teisingumas, kaip buvo aptarta aukščiau, yra vertinamas rankiniu būdu) realizuojami 1 ir 3 sakiniai. Taip turint 66% sakinio realizavimo apimtį.
  + *Antrą*, tikrinama ar AMR dalies specifikavimas yra išvis įmanomas naudojantis priemonėmis, kurias suteikia kuriamas NLG produktas.

Tokiu būdu, testavimo rezultatas susideda iš dviejų įverčių:

* 1. Teksto generavimo apimtys (angl: *scope*)
  2. Abstrakčios prasmės raiškos (AMR) apimtys

Abiejų apimčių vertinimui, sekančio projekto etapo metu, bus naudojami visi išraiškos konceptai aprašyti AMR specifikacijos [7] (<https://github.com/amrisi/amr-guidelines/blob/master/amr.md>) skyriuose:

* 1. [Part II. Concepts and relations](https://github.com/amrisi/amr-guidelines/blob/master/amr.md" \l "part-ii--concepts-and-relations),
  2. [Part III. Phenomena](https://github.com/amrisi/amr-guidelines/blob/master/amr.md" \l "part-iii--phenomena)
     + išskyrus [Non-core roles](https://github.com/amrisi/amr-guidelines/blob/master/amr.md" \l "non-core-roles) dalį.

Sekančio etapo metu siekiama turėti šias apimtis:

* 1. Teksto generavimas – 55%
  2. AMR – 40%

Kuriamas NLG produktas turi veikti el-prekybos kontekste, tačiau testavimui mes naudosime sakinių struktūras aprašytas bendriniame kontekste, taip mes užtikrinsim didesnį kuriamo produkto universalumą.

# Abstrakti prasmės reprezentacija dokumento plane

## Veiklos aprašas

2019 kovo 13 dienos ataskaitoje [1] aprašyta vartotojo sąsaja, skirta dokumento plano kūrimui, iki šiol nebuvo paremta teorija, aprašančia plano konstravimo principus. Buvo remiamasi bendresne vizualinio programavimo paradigma [7], kuri yra tinkama ir toliau naudojama, tačiau yra nepakankama aprašyti lingvistiniams objektams.

Neturint tokios dokumento plano reprezentavimo teorijos, mes negalėtume:

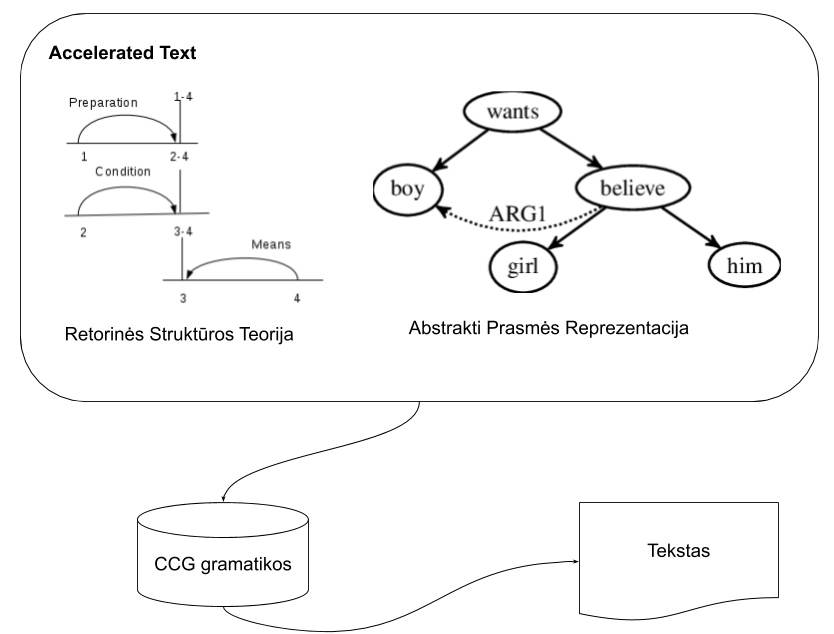
1. Išlaikyti nuoseklumą aprašant pranešimo prasminės struktūras.
2. Formalizuoti sistemos testavimą, kuriam reikalinga vienareikšmiškai aprašyti, kokie pranešimo komponentai privalo būti galutiniame tekste.

Tai nėra nauja projekto veikla. Ankstesniuose projekto etapuose, jau buvo atlikti darbai apibrėžiantys, kaip dokumento planas reprezentuos norimą pranešimą. Šio etapo metu buvo pereita prie formalizuotos ir testuojamos komponento versijos įgyvendinimo.

## Veiklos vykdymo eiga

Dokumento plano formalizavimui buvo pasirinkta Abstrakčios Prasmės Reprezentavimo (*Angl: Abstract Meaning Representaiton - AMR*) teorija [8]. Tyrimuose, atliktuose ankstesniuose projekto etapuose, mes planavome naudoti tik Retorinės Struktūros Reprezentavimo teoriją (RST) [9]. Tačiau jos abstrakcijos lygis yra per aukštas mūsų poreikiams. Tuo pačiu metu RST puikia tinka nusakyti bendrai pranešimo struktūrai. Todėl šios veiklos tikslas buvo įgyvendinti koordinaciją, tarp skirtingų prasmės aprašymo lygių:

1. RST su AMR
2. AMR su CCG
3. CCG su tekstu



*Accelerated Text* komponento pagalba, naudojantis grafine sąsaja yra kuriamos RST ir AMR reprezentacijos. Šiuo metu *Accelerated Text* leidžia aprašyti norimas prasmės reprezentacijas.

Tarkime turim tokius, turistinio bato padą, aprašančius duomenis:

|  |  |  |
| --- | --- | --- |
| **Pado medžiaga** | **Ledo danga** | **Drėgna danga** |
| Continental rubber | Taip | Taip |
| Pu | Ne | Taip |

Galimi tokie sakiniai:

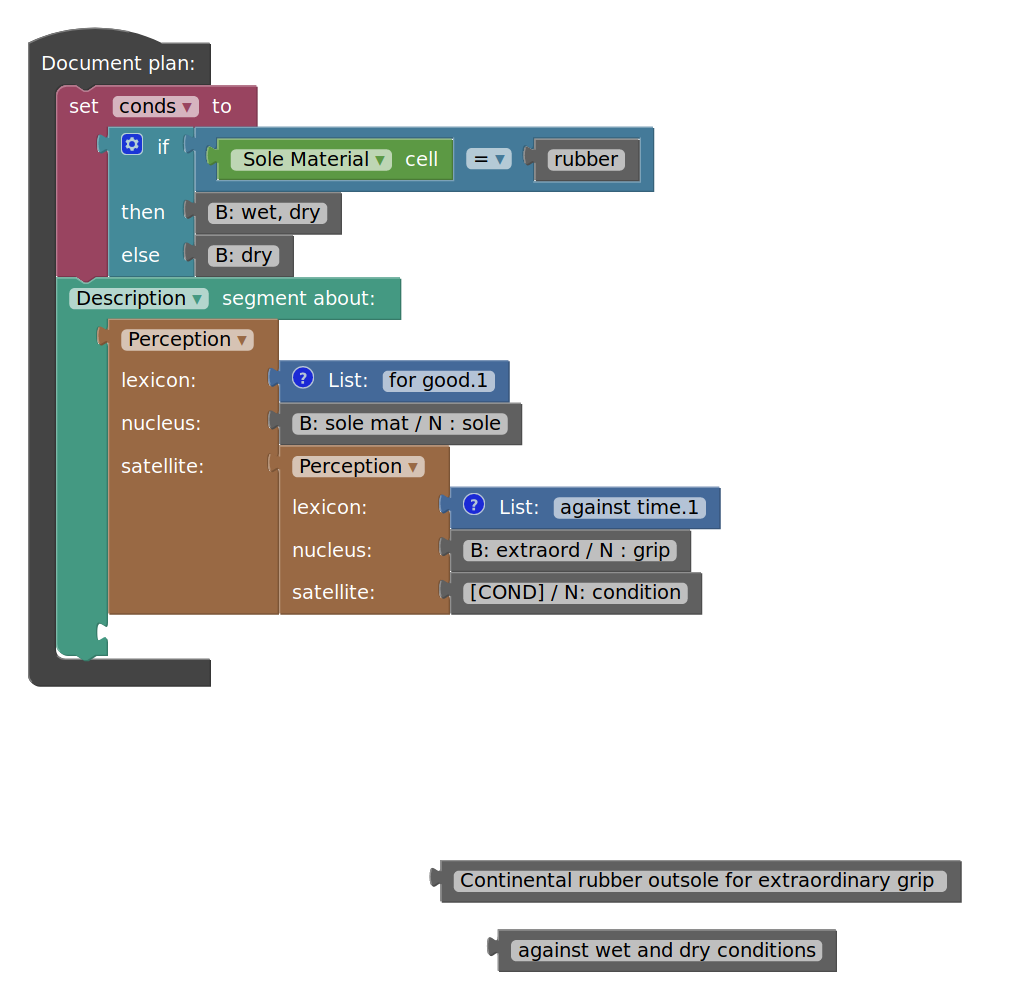
* + Contiental rubber outsole for extraordinary grip against wet and icy conditions.
  + Wet or icy conditions – continental rubber outsole will secure your grip.
  + Pu outsole provides excelent grip in wet conditions

Visais šie atvejai yra reprezentuojami ta pačia AMR struktūra, nusakančia ką teikia bato padas.

|  |  |  |
| --- | --- | --- |
| **#** | **Konceptas** | **Parametrai** |
| 1 | Sukibimas (grip) | - *Nucleus*: padas (outsole)  - *Satellite:* drėgmė *(*wet), ledas *(*icy) |
| 2 | Padas (outsole) | - *Nucleus*: *continental rubber* (outsole)  - *Satellite:* drėgmė *(*wet), ledas *(*icy) |

Čia reikia atkreipti dėmesį, kad *Nucelus*, *Satellite* parametrų prasmę suteikia *RST* [18]. Pati konceptų struktūra yra medžio pavidalo (1-oje eilutėje esantis *padas* toliau aprašomas 2-oje eilutėje).

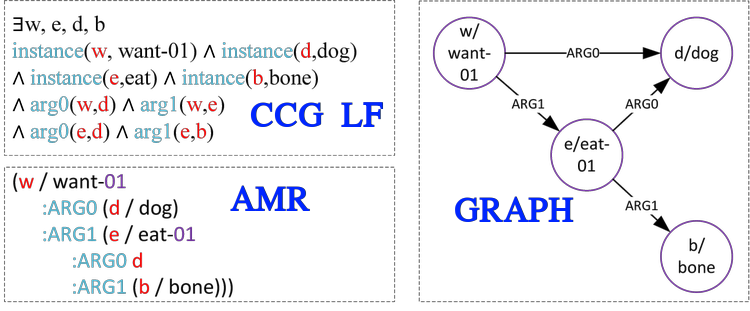
Šio plano pavyzdžio įgyvendinimas per *Accelerated Text* redaktorių, pateikimas žemiau esančiame paveiksle.

  
Pav 2: AMR plano pavyzdys

## Rezultatai

Rezultato iliustracijai imkime paprastą sakinį:

**TEXT:** The dog wants to eat the bone.

  
Pav 3: Semantinės sakinio reprezentacijos

NLG sistema, kaip detalizuota šios veiklos aprašyme, atlieka, skirtingų semantinių lygių koordinavimo ir konvertavimo, funkciją.

Tam kad gauti **TEXT** lygį, pirma turi būti sukurtas **AMR** lygis, kuris turi būti konvertuojamas į **CCG** lygį ir tada tik gaunamas tekstas. (Paveiksle esantis **GRAPH** lygis atspindi grafo reprezentacijas aprašytas ankstesnėse atskaitose).

Visi šie lygiai yra implementuoti šiuo metu veikiančioje sistemoje. Kiekvienas iš šių teksto generavimo lygmenų yra savarankiška sudėtinga sistema. Čia mes bandome apjungti kelias dideles savarankiškas informacijos perdavimo ir apdorojimo teorijas:

1. formalias gramatikas,
2. ontologijas,
3. loginį žinių reprezentavimą,
4. DI planavimą.

Visų šių komponentų koordinuotam veikimui reikia turėti ne tik gerą sprendimo architektūrą (pasirinkta ir anksčiau aprašytą *RAGS*), bet ir gerą teorinį šių sričių išmanymą. Todėl ilgalaikėje perspektyvoje šio projekto vystymas turės skirti daug daugiau dėmesio moksliniams tyrimams atitinkamose teorinėse srityse. Sekančių trijų mėnesių veiklos tikslas yra mažinti reikalingą redaktoriaus indelį kuriant reikiamus dokumentų planus, bei didinant generuojamo teksto įvairovė. Testavimo plano skyriųje (šio dokumento skyrius - 4) yra pateikiama metodologija, kaip toliau bus įgyvendinimas šis tikslas.

Rezultatus ir iššūkius, su kuriais buvo susidurta vystant šį komponentą apibendrina, žemiau pateikiama lentelė.

|  |  |  |
| --- | --- | --- |
| **Tema** | **Rezultatai** | **Iššūkiai** |
| AMR planavimas | Dokumento redaktorius leidžia apibrėžti norimas prasmės perdavimo struktūras. | - Ne visos loginės prasmės struktūros lengvai transformuojasi į gramatinę reprezentaciją  - Pats plano aprašymas kol kas per sudėtingai valdomas  - RST panaudojimas, kuris turėtų apibrėžti teksto strateginės komunikacijos lygmenį, nėra aiškiai išreiškiamas, naudojant vizualinio programavimo priemones. |
| CCG gramatikos | Įgyvendintas CCG gramatikų dinaminio generavimo variklis, kuris geba generuoti teksto variantus. | - Norint, kad CCG gramatikos padengtų visą AMR aprašomą prasmių lauką, reikia sukaupti nemažą CCG gramatikų biblioteką. Tokios bibliotekos kaupimas, gali būti daromas tik pusiau rankiniu būdu ir taip pat reikalauja nemažos lingvistinės ekspertizės. |

# Vartotojo modelio komponentas

## Veiklos aprašas

Generuojami tekstai turi varijuoti priklausomai nuo to kokiam skaitytojui tekstas yra generuojamas. Galimi du variavimo būdai:

1. **Struktūrinis** – kai teksto pranešimų seka kinta priklausomai nuo to kam yra generuojamas pranešimas. Pavyzdžiui jei kelionę į Turkiją užsisakinėja šeima, jiems generuojamas viešbučio aprašymas kalbantis apie vaikų pramogas. Jei užsisako jauna pora, generuojamas sakinys apie šalia esančius klubus.
2. **Žodyniniais** – kai atskiri žodžiai tekste yra pritaikomi prie skaitytojo. Pavyzdžiui, pirkėjų grupei kuri yra jautri kainai naudojamas *‘racionali’ kaina* išsireiškimas, kitiems tiesiog *kaina.*

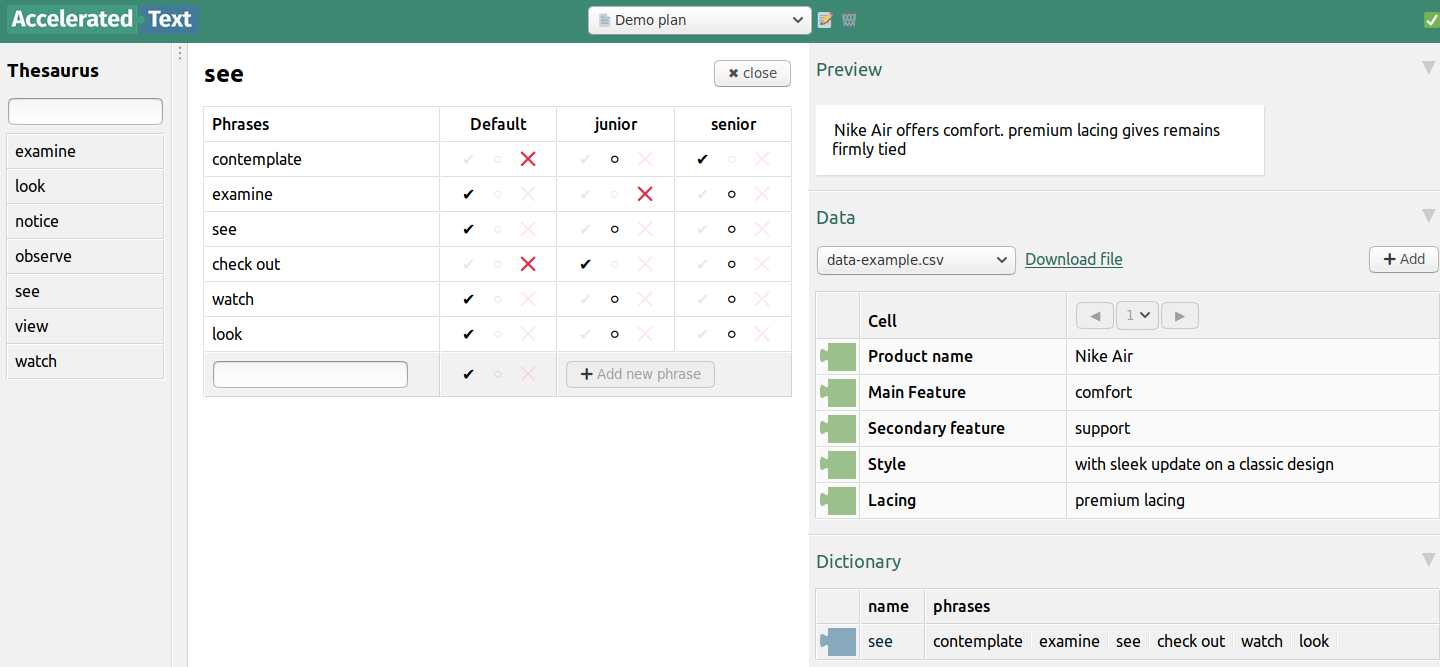
Skaitytojo (vartotojo) modelis yra perduodamas iš išorinės sistemos, dabartiniu atveju per UAB „OmniSend“ daromą vartotojų klasifikavimo modulį. NLG sistema niekaip neriboja kokie ir kiek vartotojų tipų gali būti naudojami teksto generavimo aprašymui.

## Veiklos vykdymo eiga

Šis komponentas yra vienintelis visiškai naujas komponentas įgyvendintas šiame etape. *Struktūrinis* redagavimas yra pilnai įgyvendinamas priemonėmis aprašytomis *Dokumento Plano* komponento dokumentacijoje. Žodyninis skaitytojo modelio valdymas atliekamas per įgyvendintą žodyno ir skaitytojo modelio valdymo priemonę.

## Rezultatai

Šis komponentas turi sutiekti intuityvią sąsają redaktoriui nurodyti ir/ar patvirtinti kokie žodynai ir sinonimų rinkiniai yra tinkami vienai ar kitai vartotojų grupei. Todėl turėjo būti sukurtas žodynų valdymo komponentas, kurio paveikslas pateikiamas žemiau:

  
Pav 4: Vartotojo segmento žodyno valdymo komponentas

Redaktoriui taip pat turi būti suteiktos priemonės rasti norimiems žodžiams. Žodynų sudarymui pateikiami trys šaltiniai:

1. *Rankinis* žodžių ir jų sinonimų sukūrimas;
2. *Word2Vec* [13] paremtas žodžių radimas;
3. *WordNet* [14] paremtas sinonimų rinkinių radimas.

Rezultatų ir iššūkių apibendrinimas pateikiamas žemiau esančioje lentelėje.

|  |  |  |
| --- | --- | --- |
| **Tema** | **Rezultatai** | **Iššūkiai** |
| Teksto plano modeliavimas pagal vartotojo profilį. | NLG plano redaktorius leidžia keisti teksto struktūrą pagal vartotojo modelį. | - Plano kontroliavimas pagal vartotojo tipus ženkliai komplikuoja plano redaktoriaus darbą, reikalinga toliau tobulinti ir rasti priemones lengvinančias tokių planų kūrimą |
| Teksto žodyno modeliavimas pagal vartotojo profilį. | Naujai sukurtas žodyno valdymo komponentas leidžia redaktoriui parinkti reikiamus žodžių rinkinius. | - šis komponentas itin svarbus preciziškam žodyno valdymui (neretai kaip galima kabėti apie prekę, yra apibrėžiama teisiniuose dokumentuose)  - didelio automatizavimo čia negali būti, todėl reikalingas intuityvus ir greitas, redaktoriui skirtas įrankis. |

# Galutinio teksto generavimas

## Veiklos aprašas

Kaip buvo minėta 2 Veiklos užduotys skyriuje ši veikla jau buvo pradėta pirmajame projekto etape. Galutinio sakinio generavimui tada buvo naudotas *SimpleNLG* [13]atviro kodo įrankis. Šiuo metu mes jo atsisakėme ir naudojame savo sukurtą įrankį. Atsisakymo priežastys:

1. Nepalankus SimpleNLG licencijavimas;
2. Kadangi mes kuriame *iš duomenų į tekstą* (*angl: data2text)* sistemą, mums reikalingas sakinio formos generatorius kuris užpildo šablonines duomenų vietas. Tam reikalingas specifinis realizatorius.

Tam, kad veikla būtų vykdoma greičiau mes šiuo metu implementuojame tik atskiras reikiamas realizavimo charakteristikas. Tokias kaip sakinio pabaigos skyrybos ženklai, jungtukai ir panašiai.

## Veiklos vykdymo eiga

Ši veikla reikalavo minimalių programavimo laiko investicijų ir fokusavosi į tokių sakinių realizavimą.

**CCG sakinys**: {PRODUCT} provides {BENEFITS}

**Duomenys:** PRODUCT=*Nike Air 95*; BENEFITS=*[comfort, support]*

**Sakinio realizavimas:** Nike Air 95 provides comfort **and** support**.**

Nors jungimas per ‘and’, ar ‘.’ padėjimas gale atrodo trivialus, visus šiuos sprendimus turi daryti pilnai automatizuota sistema. Toliau šis komponentas spręs ir sudėtingesnius klausimus, kaip šaukiamųjų ar klausiamųjų sakinių generavimas.

## Rezultatai

Šis komponentas vystomas pagal tokias kryptis:

|  |  |  |
| --- | --- | --- |
| **Tema** | **Rezultatai** | **Iššūkiai** |
| Sintaksės realizavimas | Šiame etape neplanuota įgyvendinti. | - neigimo konstravimas |
| Morfologinis realizavimas | Šiame etape neplanuota įgyvendinti. | - daugiskaitos ir išimčių valdymas: cat-cats, women-woman; |
| Ortografinis realizavimas | - Sakinio pradėjimas didžiąją raide;  - Elementų vardinimo jungimas per ‘,’ ar ‘ir’ konstruktus.  - Sakinio užbaigimas naudojant reikiamą skyrybos ženklą. |  |

# Įvykdyti paslaugų pirkimai

Amazon AWS serveriai ir paslaugos (dalis paslaugų šiuo metu neviršija nemokamo naudojimo limito):

1. *Amazon Lambda* – paslaugos ir NLG operacijos
2. *Amazon ES* – tekstynų saugojimas
3. *Amazon VCP* – saugumo infrastruktūra
4. *Amazon DinamoDB* – darbinių duomenų saugykla

# Užduočių atlikimas

Pateikti informaciją, kokios užduotys buvo atliktos projektą vykdančių asmenų, kas jas atliko, kiek laiko tam skyrė, naudoti lentelės formą.

| **Nr.** | **Užduotis** | **Darbuotojai/darbovietė** | **Laiko sąnaudos valandomis** |
| --- | --- | --- | --- |
| 1. | CCG analizė. Pradinių modelių įgyvendinimas | Žygimantas Medelis, UAB “TokenMill” | 168 |
| 2. | NLG sistemos architektūra ir implementacija. | Dainius Jocas UAB “TokenMill” | 431 |
| 3. | CCG gramatikos ir dokumento plano susiejimas ir konvertavimas | Šarūnas Navickas UAB “TokenMill” | 446 |
| 4. | Dokumento plano redaktorius | Tomas Čerkaukas UAB “TokenMill” | 447 |
| 5. | Kiti NLG tyrimai | Žygimantas Medelis, UAB “TokenMill” | 80 |

# Pasiekti rezultatai

Šio etapo metu buvo toliau vystomi darbai susiję su komunikacijos tikslų (dokumento plano), kurie formuluojami nepriklausomai nuo gramatinių teksto formavimo detalių, konvertavimu į gramatiškai teisingus tekstus. Būtent tekstus, nes reikalinga kurti skirtingas to paties pranešimo teksto versijas. Šioje dalyje pasiekti rezultatai:

1. CCG ir AMR formalus atskyrimas. Programinėje įrangoje dabar yra atsispindimos šios dvi teorijos. Pirmoji skirta aprašyti gramatikai, antroji pranešimo prasmei. Šis atskyrimas taip pat įgalina formalesnį testavimą.
2. Tolimesnis CCG gramatikos palaikymo vystymas, kuris mums leidžia generuoti sudėtingesnius ir didesnės įvairovės sakinius.
3. Šio etapo metu buvo sukurti baziniai AMR valdymo konstruktai. Sekančiame etape jie bus plečiami, tam, kad būtų galima valdyti platesnį prasmių perdavimo spektrą.

Teksto adaptavimo prie pasirinkto vartotojo modelio palaikymas buvo išplėstas nauju – žodynų valdymo komponentu.

Tolimesnei veiklai itin svarbus yra CCG ir AMR galimybių plėtimas. Šios srities dabai turi balansuoti tarp pranešimo tikslumo užtikrinimo ir prasmės bei gramatinių taisyklių abstraktumo išlaikymo. Pastarosioms nesant pakankamai abstrakčiomis visa sistema reikalaus per didelio redaktoriaus įsikišimo. Redaktoriaus įsikišimas (proceso valdymas) turi išlikti lankstus. Generuojant teisinius dokumentus redaktoriaus vaidmuo privalės išlikti didelis, norint užtikrinti visišką pranešimo tikslumą; generuojant drabužių prekių aprašymą toks įsikišimas gali būti mažesnis.

# Nepasiekti rezultatai

Visi numatyti rezultatai buvo pasiekti.

# Veiklos vykdymo metu atliktų tyrimų eigos pakeitimai

Veiklos pakeitimų nebuvo.

# Rezultatų naujumas ir nauda projektui

Testavimo strategijos sukūrimas atsiremiant į AMR specifikacija yra labai svarbus projekto vystymo aspektas. Tai taip pat yra ir nauja NLG sistemų testavimo metodologija, kuri nesiremia, ne visai tinkamais, iš mašininio vertimo perimtais, metodais (BLEU).

Loginės pranešimo struktūros ir generavimo gramatikos susiejimas taip pat yra svarbus ir NLG sistemose retai naudojamas sprendimas. Kaip aprašoma *Blodgett A. (2019) [17] ir Artzi Y. (2015) [16]* tokia technika leidžia išvengti giliojo mokymo (angl: *deep learning*) stiliaus apmokymų ant milžiniškų duomenų kas nei yra prieinama mūsų kontekste, nei užtikrina reikiamą teksto tikslumą.

# Bibliografija

1. 2019-03-13 dienos ataskaita
2. 2018-12-13 dienos ataskaita
3. *Clojure* (Skaityta: 2019-06-07) https://clojure.org/
4. *Datomic* (Skaityta: 2019-06-07) <https://www.datomic.com/>
5. *RESTful* (Skaityta: 2019-06-07), https://en.wikipedia.org/wiki/Representational\_state\_transfer
6. *GraphQL* (Skaityta: 2019-06-10), https://graphql.org/
7. *Abstract Meaning Representation,* (Skaityta: 2019-06-10) <https://amr.isi.edu/>
8. C. Bozşahin*,* et al*, Specifying Grammars for OpenCCG:A Rough Guide* (2005) https://users.metu.edu.tr/bozsahin/nli/ceng563/link/grammars-rough-guide.pdf
9. *BLEU*, <https://en.wikipedia.org/wiki/BLEU> (Skaityta: 2019-06-10)
10. Chaganty A. T., Mussman S., Liang P. *The price of debiasing automatic metrics in natural language evaluation* (2018) <https://arxiv.org/abs/1807.02202>
11. *AMR specification* (Skaityta: 2019-06-11) https://github.com/amrisi/amr-guidelines/blob/master/amr.md
12. 2018-09-25 dienos projekto ataskaita
13. *SimpleNLG* (Skaityta: 2019-06-11) <https://github.com/simplenlg/simplenlg>
14. *Word2vec* (Skaityta: 2019-06-12) https://en.wikipedia.org/wiki/Word2vec
15. *WordNet* (Skaityta: 2019-06-13) https://wordnet.princeton.edu/
16. Artzi Y. et al, *Broad-coverage CCG Semantic Parsing with AMR* (2015) <https://yoavartzi.com/pub/alz-emnlp.2015.pdf>
17. Blodgett A., Schneider N., *An Improved Approach for Semantic Graph Compositionwith CCG* (2019) <http://people.cs.georgetown.edu/nschneid/p/ccgamr.pdf>
18. *Rhetorical Structure Theory* (Skaityta: 2019-06-12) <https://www.sfu.ca/rst/>