VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
INFORMACINIŲ SISTEMŲ KATEDRA

Igor Marin

KELIŲ AUTOMATINIO VERTIMO SISTEMŲ INTEGRACIJA

THE INTEGRATION OF SEVERAL AUTOMATIC TRANSLATION SYSTEMS

Baigiamasis magistro darbas

Inžinerinės informatikos studijų programa, valstybinis kodas 621I13002
Informacinių sistemų specializacija
Informatikos studijų kryptis

Vilnius, 2012

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
INFORMACINIŲ SISTEMŲ KATEDRA

TVIRTINU
Katedros vedėjas

_____
(Parašas)

__Dalius Mažeika_____
(Vardas, pavardė)

_____
(Data)

Igor Marin

KELIŲ AUTOMATINIO VERTIMO SISTEMŲ INTEGRACIJA

THE INTEGRATION OF SEVERAL AUTOMATIC TRANSLATION SYSTEMS

Baigiamasis magistro darbas

Inžinerinės informatikos studijų programa, valstybinis kodas 621I13002
Informacinių sistemų specializacija
Informatikos studijų kryptis

Vadovas _____dr. Algirdas Laukaitis_____ _____ _____
(Moksl. laipsnis/pedag. vardas, vardas, pavardė) (Parašas) (Data)

Konsultantas _____ _____ _____
(Moksl. laipsnis/pedag. vardas, vardas, pavardė) (Parašas) (Data)

Lietuvių kalbos konsultantas_____dr. Angelė Kaulakienė__ _____ _____
(Moksl. laipsnis/pedag. vardas, vardas, pavardė) (Parašas) (Data)

Vilnius, 2012

Vilniaus Gedimino technikos universitetas
Fundamentinių mokslų fakultetas
Informacinių sistemų katedra

ISBN ISSN
Egz. sk. ……..
Data ….-….-….

Inžinerinės informatikos studijų programos baigiamasis magistro darbas

Pavadinimas **Kelių automatinio vertimo sistemų integracija**

Autorius **Igor Marin** Vadovas doc. dr. **Algirdas Laukaitis**

Kalba
[X] lietuvių
[ ] užsienio

Anotacija

Baigiamajame magistro darbe nagrinėjamos automatinės vertimo sistemos, pagrindiniai jų veikimo principai ir šių sistemų integracijos būdai. Detaliai aprašoma populiarių šiuo metu statistinių vertimo sistemų struktūra, pateikiami šių ir tradicinių (taisyklėmis paremtų) sistemų privalumai ir trūkumai. Pristatomos visos šiuo metu egzistuojančios automatinio vertimo sistemos lietuvių ir anglų kalbų porai, išaiškinami jų privalumai ir trūkumai. Lingvistiniu požiūriu nagrinėjamos lietuvių ir anglų kalbos, išvardinami šių kalbų panašumai, skirtumai ir sunkumai, kylantys verčiant iš vienos kalbos į kitą. Taip pat pateikiami įvairūs automatinio vertimo įvertinimo būdai, įskaitant populiarų *BLEU* įvertinimo metodą. Išvardinamos ir analizuojamos užsienio autorių siūlomos automatinio vertimo sistemų integracijos architektūros. Apžvelgiami sumaišymo tinklai, kurie naudojami kuriant integruotą vertimo sistemą. Pateikiama originali mišriosios vertimo sistemos įgyvendinimo metodika. Integruota sistema yra praktiškai įgyvendinama. Šios sistemos ir kitų vertimo sistemų anglų ir lietuvių kalbų porai rezultatai yra įvertinami ir palyginami. Atlikus teorinę automatinio vertimo sistemų apžvalgą ir praktiškai įgyvendinus mišriąją vertimo sistemą, pateikiamos baigiamojo darbo išvados ir siūlymai.

Darbą sudaro 6 dalys: įvadas, automatinio vertimo sistemų analizė, mišriųjų automatinio vertimo sistemų analizė, mišriosios automatinio vertimo sistemos sukūrimas, išvados, literatūros sąrašas.

Darbo apimtis – 60 p. teksto be priedų, 17 iliustr., 9 lent., 33 bibliografiniai šaltiniai.

Atskirai pridedami darbo priedai.

Prasminiai žodžiai: automatinis vertimas, statistinis vertimo metodas, mišrioji vertimo sistema, sistemų integracija, sumaišymo tinklas, *BLEU* metrika, n-grama.

Vilnius Gediminas Technical University
Faculty of **Fundamental Sciences**
Department of **Information Systems**

ISBN ISSN
No. of copies ……..
Date ….-….-….

Master's thesis in **Engineering Informatics**

Title **The integration of several automatic translation systems**

Author **Igor Marin** Supervisor assoc. prof. dr. **Algirdas Laukaitis**

Language
[X] Lithuanian
[ ] Foreign (English)

Summary

The Master's thesis analyses machine translation systems, their principles of operation and the methods used in integrating these systems. The structure of popular statistical machine translation systems, as well as the advantages and disadvantages of such systems is described in detali. The existing machine translation systems for the Lithuanian-English language pair along with their abilities and shortcomings are presented. Lithuanian and English languages are analysed from the linguistic perspective. The similarities and differences between these languages, as well as the difficulties, arising in translating the text from one language to another are discussed. Moreover, different machine translation evaluation methods, including the popular *BLEU* metric, are reviewed. Various architectures for integrating multiple machine translation systems, offered by foreign authors, are presented and analysed. Confusion networks, which are used in integrating machine translation systems, are discussed. An original method of implementing the hybrid machine translation system is offered. The hybrid system is implemented in practice. The translation results obtained from the created system and the existing systems for the Lithuanian-English language pair are assessed and compared. Finally, after performing the theoretical review of machine translation systems and implementing the hybrid system, conclusions and recommendations are provided.

The thesis consists of 6 parts: introduction, the analysis of machine translation systems, hybrid machine translation systems, the creation of a hybrid machine translation system, conclusions, references.

The size of thesis is 60 p. without appendices, 17 figures, 9 tables, 33 references.

The appendix to the thesis is provided.

Keywords: machine translation, statistical translation method, hybrid translation system, system integration, confusion network, *BLEU* metric, n-gram.

Paveikslų ir lentelių sąrašas

Paveikslai

Lentelės

Turinys

Įvadas

Globalizacijos metu ypač stiprėja keitimasis informacija, todėl didėja vertimo ir vertėjų poreikis. Yra didelis poreikis susipažinti su juridine, technine ir kitokia informacija, o taip pat supažindinti pasaulį su Lietuvos pasiekimais, ypač mokslininkų darbais, ekonominio vystymosi rezultatais, kultūriniu gyvenimu ir t.t. Esant šioms sąlygoms, žymiai išaugo automatinio (kompiuterinio, mašininio) vertimo vaidmuo, kuris, kaip žinoma, nėra tobulas. Automatinis vertimas išlieka viena aktualiausių natūralių kalbų apdorojimo (angl. *Natural Language Processing* arba *NLP*) srities problemų. Tai rodo reguliarūs automatinio vertimo įvertinimai, atliekami amerikiečių nacionalinio standartų ir technologijos instituto (angl. *National Institute of Standards and Technology* arba *NIST*), kurie leidžia teigti, kad automatinis vertimas tobulėja ir dar labiau pritraukia naujų mokslininkų dėmesį. Pastarųjų penkių dešimtmečių automatinio vertimo tyrimų rezultatai leido atsirasti daugybei vertimo sistemų, turinčių kiek privalumų, tiek trūkumų. Reikia pažymėti, kad autorius gerai moka anglų kalbą ir turėdamas techninių vertimų praktikos supranta lingvistinius vertimo sunkumus, o programavimo patirtis leidžia autoriui suprasti ir techninę vertimo programų pusę. Praeitame autoriaus darbe (Marin 2010) buvo išsiaiškinta, kad vertimo sistemos yra nepajėgios susidoroti su kalbos daugiareikšmiškumu arba polisemija, todėl vertimas nėra tikslus ir reikalauja patyrusių vertėjų postredagavimo. Šio **darbo tikslas** yra pasiūlyti mišriosios automatinio vertimo sistemos variantą, kuris pateiktų tikslesnį vertimą lietuvių ir anglų kalbų porai. Kitaip tariant, yra bandoma patikrinti hipotezę, kad kelių lietuvių ir anglų kalboms skirtų automatinio vertimo sistemų integracija pranoksta pavienių vertimo sistemų rezultatus. Šiam tikslui pasiekti iškelti **uždaviniai**:

- atlikti egzistuojančių automatinio vertimo sistemų analizę;
- parodyti kelių automatinio vertimo sistemų integracijos (mišriosios automatinio vertimo sistemos) perspektyvumą argumentuojant darbo temos pasirinkimą;
- išnagrinėti kelių automatinio vertimo sistemų integracijos metodikas;
- išnagrinėti mišriųjų vertimo sistemų architektūras, jų komponentus, šių komponentų sąryšius ir kiekvieno iš jų įtaką galutiniam vertimui;
- išnagrinėti programinius įrankius, naudojamus praktiniam mišriųjų vertimo sistemų įgyvendinimui;
- pateikti ir pagrįsti vienos iš esančių mišriųjų vertimo sistemų architektūrų modifikacijas;
- remiantis patobulinta architektūra ir naudojant atvirojo kodo statistinio vertimo įrankį *Moses* ir kitokius įrankius pasiūlyti mišriosios vertimo sistemos variantą;

- visapusiškai įvertinti pasiūlytas automatinio vertimo sistemas (naudojant *BLEU* vertimo vertinimo metriką gautus rezultatus palyginti su pavienėmis vertimo sistemomis bei kitomis mišriosiomis sistemomis);
- apibendrinti gautus rezultatus, pateikti tolesnes sukurtos sistemos plėtojimo perspektyvas.

Darbas yra **aktualus** ir **naujoviškas**, kadangi kompiuterinio vertimo poreikis nuolat auga, bet šiuo metu egzistuojantys vertimo metodai ir programos reikalauja tobulinimo. Be to, Lietuvoje trūksta automatinio vertimo sistemų, verčiančių iš lietuvių į anglų kalbą, kas yra ypač aktualu norint teikti informaciją apie šalies gyvenimą užsieniui. Tokio tipo sistemos sukūrimas, numatytas šiame darbe, ir jos veikimo demonstravimas ir jos rezultatų tikslumo palyginimas su jau egzistuojančių sistemų efektyvumu turi **praktinę vertę**. Darbo **teorinę vertę** sudaro egzistuojančių sistemų privalumų ir trūkumų analizė, sistemų vertimo rezultatų analizė ir pasiūlyta originali šių sistemų integracijos įgyvendinimo koncepcija.

Pirmajame skyriuje apibūdinami pagrindiniai automatinio vertimo sistemų veikimo principai. Detaliai nagrinėjamas dominuojantis šiuo metu statistinio vertimo principas. Toliau yra aprašomi automatinio vertimo sistemų veikimo principų privalumai ir trūkumai. Ketvirtajame skyriaus poskyryje pateikiama lietuvių ir anglų kalbų analizė lingvistiniu požiūriu. Kitame poskyryje analizuojamos šiai kalbų porai skirtos automatinės vertimo sistemos, įskaitant ir *Moses*, kuri buvo panaudota kuriant naują integruotą vertimo sistemą. Skyrių pabaigia informacija apie automatinio vertimo įvertinimą. Antrajame skyriuje yra argumentuojama, kad kelių sistemų integracija padeda pasiekti tiksliausio teksto vertimo. Be to, yra aprašomos mišriųjų vertimo sistemų metodikos ir architektūros. Paskutiniame skyriuje apibūdinama metodika, panaudota mišriajai vertimo sistemai sukurti ir aprašomas sistemos veikimas. Galiausiai yra suformuluojamos išvados ir pateikiami pasiūlymai.

1. Automatinio vertimo sistemų analizė

Koks yra automatinio vertimo sistemų veikimo principas? Egzistuoja du pagrindiniai principai, naudojami kuriant mašininio vertimo programas: vienos sistemos yra pagrįstos taisyklėmis (angl. *rule-based machine translation*), pavyzdžiui, *Promt* ir *Systran*, kitos – statistika (angl. *statistical-based machine translation*), pavyzdžiui, *Google Translate* sistema. Ir taisyklėmis pagrįstos, ir statistinės vertimo sistemos turi tiek privalumų, tiek trūkumų. Todėl, siekdami išnaudoti abiejų išvardytų sistemų tipų privalumus, lingvistikos mokslininkai nusprendė sujungti šias sistemas į vieną mišrią vertimo sistemą (angl. *hybrid machine translation*). Geriausius vertimo rezultatus turėtų pateikti abiem metodais pagrįsta kompiuterinio vertimo sistema.

1.1. Taisyklėmis pagrįstos automatinio vertimo sistemos

Taisyklėmis paremtos vertimo sistemos pagrindą sudaro didžiulis skaičius lingvistinių taisyklių ir milijonai kalbinių porų įrašų. Tokia sistema analizuoja tekstą ir remdamasi verčiamo sakinio analize sintezuoja vertimo variantus. Toks darbas gali būti palygintas su žmogaus mąstymu: vertimo sistema analizuoja tekstą naudodama daugybę algoritmų, panašiai kaip ir žmogus analizuoja verčiamą tekstą (Rimkute *et al.* 2007).

Taisyklėmis pagrįstos automatinio vertimo sistemos darbas skirstomas į kelis etapus: pirmiausia vyksta morfologinė analizė, t. y. žodžiai priskiriami gramatinėms kategorijoms, tokioms kaip kalbos dalis, giminė, skaičius, asmuo ir t.t. Šitame analizės etape programa nesprendžia morfologinio daugiareikšmiškumo problemos. Antrasis etapas – susijusių žodžių grupių analizė ir sintezė. Šitame etape iš kelių morfologiškai daugiareikšmių formų pasirenkama viena; žodžiai sujungiami į grupes: veiksmažodines (pvz., *have taken*), daiktavardines (pvz., *a woman*) ir pan. Trečiasis etapas – sintaksinė sakinio analizė: nustatomos sakinio dalys, vientisinių sakinių ribos, vientisinių sakinių ryšys sudėtiniame sakinyje. Pirmiausia programa ieško tarinio, paskui veiksnio. Jei nerandamas veiksnys, daroma išvada, kad sakinyje pavartotas liepiamosios nuosakos arba beasmenis veiksmažodis. Paskutinis etapas – tos kalbos, į kurią verčiama, sakinio sintezė: tarpusavyje suderinami frazes sudarantys žodžiai, tarinys ir kitos nuo jo priklausančios sakinio dalys, patikslinama žodžių tvarka (Андреев 2007).

1.2. Statistinės automatinio vertimo sistemos

Statistinis vertimo metodas yra duomenimis pagrįstas (angl. *data-driven*) metodas ir šiuo metu populiariausias. Paskutiniai lyginamieji įvertinimai parodė, kad statistinės automatinio vertimo sistemos

jau pradeda lenkti taisyklėmis pagrįstas sistemas. Be to, statistinių sistemų kūrimo sunkumas žymiai sumažėjo, pasirodžius atvirojo kodo įrankiams, tokiems kaip *GIZA++* ir *Moses*.

Statistinis metodas nenaudoja lingvistinių algoritmų, o yra pagrįstas statistiniu galimų tikimybių skaičiavimu. Šiuo metodu pagrįstos vertimo sistemos verčia dokumentą remdamosios tikimybių skirstiniu *p(e|f)* tokiu, kad eilutė *e* kalboje, į kurią verčiama, yra lygi eilutei *f* pradinėje kalboje. Tikimybinis skirstinys gaunamas apdorojus didelius dvikalbius lygiagrečius tekstynus. Vienoje rinkmenoje saugomas pirmasis tekstynas, tai yra visi viena kalba sukaupti sakiniai, o kitoje – antrasis tekstynas su ta pačia tvarka kaip ir pirmajame surūšiuotais sakiniais, tik parašytais kita kalba.

Tikimybių skirstinio *p(e|f)* modeliavimo problemą yra bandoma spręsti įvairiais būdais, pavyzdžiui, naudojant Bejeso teoremą. Statistinės vertimo sistemos turi atlikti efektyvią paiešką visų pradinės kalbos eilučių *f** rinkinyje, todėl ji turi specialų automatinio vertimo dekoduotoją, kuris naudoja euristinius ir kitokius metodus paieškos sričiai apriboti ir kokybiškam vertimui pateikti.

Iš pradžių statistinės vertimo sistemos versdavo pavienius žodžius, tačiau dabar galimas frazių, sakinių ir pastraipų vertimas.

1.2.1. Žodžiais pagrįstas automatinis vertimas

Pirmiausias ir pats paprasčiausias statistinio vertimo variantas – žodžiais pagrįstas statistinis vertimo metodas. Oficialiai jis atsirado 1993 m., kai pasirodė žymus *IBM* kompanijos tyrėjų Browno ir kt. straipsnis (Brown *et al.* 1993) apie statistinius metodus, kuriuos galima panaudoti atliekant automatinį vertimą. Autoriai statistiniam vertimui pritaikė garsiąją Bejeso teoremą:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}; \tag{1}$$

čia *P(A|B)* – sąlyginė tikimybė (A tikimybė, esant B), *P(B|A)* – sąlyginė tikimybė (B tikimybė, esant A) arba tikėtinumas, *P(A)* – apriorinė arba besąlyginė A tikimybė, *P(B)* – apriorinė arba besąlyginė B tikimybė ir normalizuojanti konstanta. Brownas ir kt. savo darbe Bejeso teoremą išreiškė tokiu būdu:

$$\hat{e} = \arg\max_e p(f \mid e)p(e); \tag{2}$$

čia $\hat{e}$ – dekodavimo funkciją argmax maksimizuojantis argumentas (vertimas) *e*. Būtent ši formulė išreiškia fundamentalią statistinio vertimo srities lygtį. Sąlyginė tikimybė *p(e | f)* arba $\hat{e}$ yra suskaidoma į ***vertimo modelį*** *p(f | e)* ir ***kalbos modelį*** *p(e)*. Tai panašu į kriptografijoje egzistuojantį ***šaltinio ir kanalo modelį***, kur *p(e)* yra šaltinio sklidimas, *p(f | e)* yra užkoduojančio (iškraipančio) proceso modelis, o *argmax* yra dekodavimo operacija. Toks suskaidymas duoda galimybę dirbti su vienkalbiu tekstynu ir tobulinti kalbos modelį.

Brownas ir jo kolegos taip pat pasiūlė penkis *p(f | e)* vertimo modelius, vadinamus *IBM* modeliais. Kiekvienas kitas modelis yra pranašesnis už ankstesnįjį tuo, kad įvedami vis nauji ar kitaip interpretuojami seni parametrai. *IBM* modeliai yra apmokomi tokiu būdu, kad maksimizuotų lygiagretaus dvikalbio tekstyno, suprantamo kaip statistiškai nepriklausomų sakinių porų, tikėtinumą. Ankstesnieji modeliai pateikia pradinių parametrų įverčius kitiems modeliams. *IBM* modeliai susiduria su optimalaus vertimo paieškos (argmax operacija formulėje (2)) problema, kadangi yra sudėtinga surasti vertimo žodžių rinkinį esant *p(f | e)* šaltinio ir tikslo ryšiui „daug-su-vienu“ ir geriausią vertimo žodžių seką pagal *p(e)*. Amerikietis Knightas savo darbe (Knight 1999) parodo, kad *IBM* modeliuose ši problema įgyja *NP* pilnumo laipsnį, tai yra surasti sprendimą praktiškai neįmanoma. Todėl praktikoje yra naudojami euristiniai metodai, kurie padeda surasti apytikslį sprendimą. Vienas iš tokių metodų pavyzdžių yra Viterbio algoritmas, pasižymintis greitaveika ir padarantis minimalų paieškos klaidų skaičių.

IBM modeliai buvo išstumti naujesniu frazėmis pagrįstu statistinio vertimo būdu, kuris yra paprastesnis ir pateikia geresnius rezultatus. Tačiau jie vis dar yra naudojami dėl jų gebėjimo pateikti gerus dviejų kalbų žodžių atitikimo grafus, kurie yra frazėmis pagrįstų modelių pagrindas. *IBM* 4 modelio grupavimai (dažniausiai atlikti su *GIZA++* įrankiu), simetrizuoti panaudojant Ocho ir Ney metodą, išlieka dažniausiai naudojamu išeities tašku projektuojant statistines vertimo sistemas (Cancedda *et al.* 2009).

1.2.2. Kalbos modeliai

Kalbos modelis, paprastai tariant, yra apskaičiuojamas tikimybinis žodinių sekų (paprastai sakinių) skirstinys, gaunamas remiantis išanalizuotu tekstynu. Kiekvienam teksto fragmentui priskiriama tam tikra tikimybė remiantis fragmento žodžių pasitaikymo tekstyne dažnumu.

Kalbos modeliai, išskyrus statistinį vertimą, yra taikomi daugybėje kitų sričių, tokių kaip kalbos atpažinimas, rašybos klaidų taisymas, optinis simbolių atpažinimas, informacijos išgavimas ir kt. Kalbos modelių plėtotė daugiausia susijusi su kalbos atpažinimu, tačiau pirmuosius kalbos modelius (vadinamus Markovo modeliais) panaudojo rusų matematikas Andrejus Markovas, tirdamas raidžių sekas literatūros kūriniuose.

Dominuojantis metodas formuojant kalbos modelį yra n-gramų metodas, kuriuo galima gauti apytikres žodžių sekų w_1 , w_2, . . . , w_m tikimybes. n-gramų metodo esmę sudaro ribotas praeities būsenų, o tiksliau n-1 žodžių, skaičius iš prieš tai buvusio konteksto. Todėl, pavyzdžiui, bigramos yra pirmosios eilės, o trigramos – antrosios eilės Markovo modeliai ir t.t. Trigramų atveju turime:

$$p(w_1, w_2, ..., w_m) \approx \prod_i p(w_i \mid w_{i-2}, w_{i-1}) \,. \qquad (3)$$

Pagrindinis sunkumas yra sąlyginių tikimybių p (w_i | w_{i-2}, w_{i-1}) įvertinimas remiantis tekstynu. Paprasčiausias būdas yra tikėtinumo maksimizavimo metodas, kuris įvertina šias tikimybes kaip santykį kartų, kai tekstyne yra aptinkamos atitinkamos n-gramos:

$$p(w_i \mid w_{i-2}, w_{i-1}) = \frac{\#(w_{i-2}, w_{i-1}, w_i)}{\#(w_{i-2}, w_{i-1})} . \quad (4)$$

Tačiau toks būdas nepasiteisina. Pavyzdžiui, modelis priskiria nulinę tikimybę trigramai, kuri nė karto nebuvo aptikta tekstyne. Norint išspręsti šią problemą yra naudojamas tikimybių glotninimas, kuris reiškia, kad nematytoms tekstyne n-gramoms bus priskiriama nenulinė tikimybė. Egzistuoja įvairūs būdai, kaip tai padaryti, pavyzdžiui, tai gali būti interpoliacinis glotninimas, kur panaudojamos žemesnio lygio n-gramos su glotninimo koeficientais, arba grįžimo atgal metodai, kai esant nulinėms tikimybėms pereinama prie žemesnio lygio n-gramų skaičiavimo (pvz., trigramų atveju, jei kai kurios trigramų tikimybės nulinės, pereinama prie bigramų modeliavimo).

Dar vienas būdas tikimybių glotninimui yra klasėmis paremtas glotninimas. Vietoje žodžiais paremto glotninimo galima suskirstyti visus žodžius į klases, kurios pasižymi panašiomis lingvistinėmis savybėmis ir tada naudoti šias klases statistinių priklausomybių modeliavimui.

$$p(w_i \mid w_{i-2}, w_{i-1}) \approx p(w_i \mid C_i)\, p(C_i \mid w_{i-2}, w_{i-1}) ; \quad (5)$$

čia C_i – klasė, kuri yra asocijuojama su w_i. Toks modelis naudingas, nes klasės tekstyne aptinkamos dažniau nei individualūs žodžiai, todėl sąlyginės tikimybės, panaudojančios klases, gali būti įvertintos patikimiau. Klasės gali būti naudojamos įvairiose kombinacijose (kairėje ar dešinėje sąlygos ženklo pusėje). Be to, vienam žodžiui gali būti priskiriama viena arba kelios klasės pagal iš anksto aprašytas kategorizacijos schemas, tokias kaip kalbos dalių žymekliai (angl. *part-of-speech tags*).

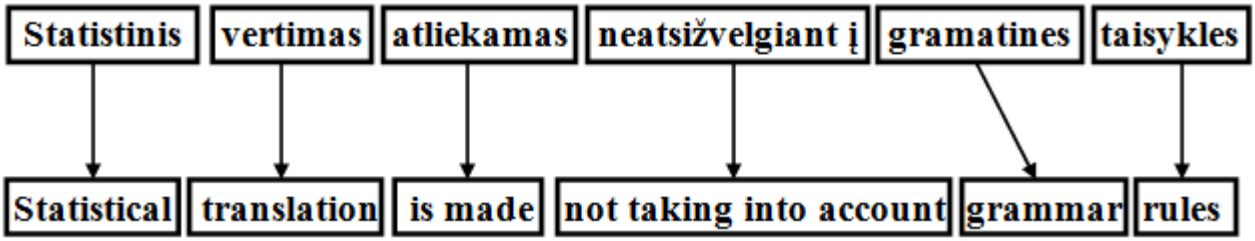
1.2.3. Frazėmis pagrįstas automatinis vertimas

Frazėmis pagrįstas statistinis automatinis vertimo būdas šiuo metu yra dominuojantis. Jis panaudoja penkias naujoves, dėl kurių jis yra daug pranašesnis už žodžiais paremtą vertimo būdą:

- logtiesiniai modeliai vietoje paprastos kalbos ir vertimo modelių sandaugos;
- daugiažodžių frazių kaip bazinio vertimo vieneto naudojimas paprastesniame „vienas-su-vienu“ ryšį turinčiame vertimo modelyje;
- logtiesinių modelių minimalaus klaidų dažnumo apmokymas vietoje tikėtinumo maksimizavimo apmokymo (žodžiais pagrįsto vertimo atveju);
- aiškiai apibrėžta ir naši euristinė Viterbio spindulio paieškos procedūra;

- naudojamas antrasis įvertinimo etapas tam, kad būtų galima surasti geriausią hipotezę iš mažo kandidatų rinkinio, pateikto paieškos metu.

Frazėmis pagrįstas vertimo metodas suskaido įvestą sakinį į frazių seką. Gautos frazės yra susiejamos ryšiais „vienas-su-vienu“ su išvesties (vertimo) frazėmis (1 pav.). Kiekvienai paduoto sakinio frazei parenkamas geriausias atitikmuo ir tokie surikiuoti atitikmenys sudaro *e* (išverstą tekstą).



1 pav. Frazėmis pagrįsto vertimo metodas, lietuviško sakinio frazių priskyrimas angliško sakinio frazėms

1.2.3.1. Logtiesiniai modeliai

Kaip paminėta anksčiau, šaltinio ir kanalo metodas sujungia kalbos modelio *p(e)* ir atvirkštinio vertimo modelio *p(f | e)* rezultatus juos padaugindamas. Siekiant išskirti skirtingą šių modelių svarbą, galima kiekvienam iš jų pritaikyti eksponentinius svorius $p(e)^{\alpha}$ $p(f \mid e)^{\alpha}$. Logaritmuojant gaunamas modelių aproksimavimas tiesinių funkcijų *h(f,e)* pavidalu arba logtiesinis formulės (2) pavidalas:

$$\hat{e} = \arg\max_{e} \sum \alpha_i h_i(f,e) \approx \arg\max_{e,a} \sum_i a_i h_i(f,a,e). \quad (6)$$

Čia pademonstruota standartinė Viterbio aproksimacija, supaprastinanti paieškos problemą ir panaudojanti parametrą *a*, kuriame saugomas *f* (kalbos iš kurios verčiamos frazės) ir *e* (kalbos, į kurią verčiamos frazės) ryšis. Toks metodas yra lankstesnis nei duomenų šaltinio ir kanalo metodas, nes jam lengviau pritaikomi dvikalbiai žodynai, kuriuos sunku integruoti į tikimybinius vertimo modelius.

1.2.3.2. Frazėmis pagrįsto vertimo modelis

Frazinio vertimo modeliai gaunami iš lygiagretaus tekstyno, kurio vienos kalbos žodžiai buvo priskirti kitos kalbos žodžiams. Į modelį įtraukiamos visos frazių poros, kurios atitinka žodžių priskyrimo (angl. *word alignment*) rezultatus. Tikimybiniai įverčiai priskiriami pagal kiekį kartų, kuriuos atitinkamos frazės buvo aptiktos lygiagrečiame tekstyne. Frazių vertimo tikimybių skirstiniai kartu su atitinkamomis frazių poromis sudaro frazių lentelę (angl. *phrase table*).

Naujausi frazių lentelių išgavimo iš lygiagretaus tekstyno būdai neišvengiamai prasideda nuo žodžių priskyrimo. Pats populiariausias įrankis šiai operacijai atlikti yra *GIZA++*. Šis įrankis yra originalių *IBM* modelių, pradėjusių statistinio vertimo tyrimus, realizacija. Tačiau kadangi šie modeliai turėjo keletą didelių trūkumų (svarbiausias iš jų – vienas vienos kalbos žodis gali būti priskirtas tik vienam kitos kalbos žodžiui), *GIZA++* buvo žymiai patobulintas.

Prieš atliekant žodžių priskyrimą susiduriama su problema: „Kokį kitos kalbos žodį atitinka pasirinktas šaltinio kalbos žodis?“ Šiam klausimui atsakyti naudojamas jau minėtas tikėtinumo maksimizavimo arba *EM* (angl. *expectation maximization*) metodas. *EM* algoritmas – tai statistinis iteracinis mokymosi metodas, kuris nustato maksimaliai tikėtinas numanomų parametrų reikšmes. Plačiau su algoritmu galima susipažinti (Koehn 2009: 88-93). Žemiau pateiktoje lentelėje parodoma algoritmo esminė idėja – tekstyno žodžių priskyrimo vienas kitam kartų skaičiavimas.

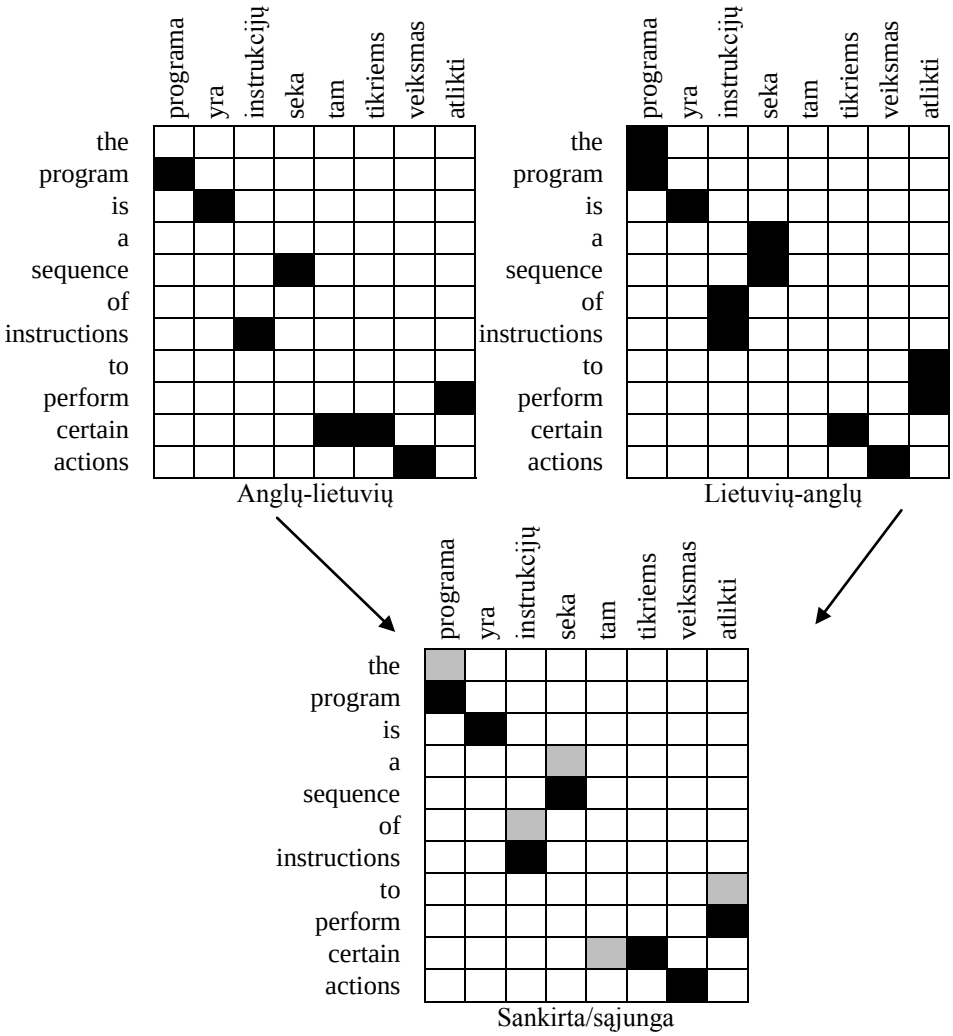
1 lentelė. Trijų sakinių dvikalbis tekstynas ir jo žodžių atitikmenų paieška

Tekstyno turinys					
Lietuvių	Kompiuteris yra galingas				
Anglų	The computer is powerful				
Lietuvių	Variklis yra galingas				
Anglų	The engine is powerful				
Lietuvių	Galingas kompiuteris yra brangus				
Anglų	A powerful computer is expensive				
Žodžių atitikmenų paieška					
	kompiuteris	yra	galingas	variklis	brangus
the	1	2	2	1	
computer	2	2	2		1
is	2	3	3	1	1
powerful	2	3	3	1	1
engine		1	1	1	
a	1	1	1		1
expensive	1	1	1		1

Lentelėje pateiktas tekstynas iš trijų paprastų sakinių. Po sakiniais parodomi kompiuterio, veikiančio pagal *EM* algoritmą, rezultatai. Kadangi nežinome kokį anglišką žodį atitinka kiekvienas lietuviškas žodis tikriname kiekvieną įmanomą variantą. Pirmiausiai pirmo sakinio lietuviškiems žodžiams „kompiuteris“, „yra“ ir „galingas“ priskiriamas artikelis „*the*“. Tuomet tiems patiems žodžiams priskiriamas žodis „*computer*“ ir t.t. Tokie patys veiksmai atliekami su antro ir trečio sakinių žodžiais. Galiausiai galima pamatyti, kad angliškas žodis „*is*“, kaip ir lietuviškas žodis „yra“, yra kiekviename tekstyno sakinyje, todėl labai tikėtina, kad šie žodžiai turi tą pačia reikšmę. Tekstyno dydis yra per mažas, todėl kompiuteris su ta pačia tikimybe taip pat gali teigti, kad „*is*“ vertimas yra žodis „galingas“. Esant pakankamai dideliam sakinių kiekiui pastarasis žodis nėra aptinkamas taip dažnai kaip žodis „yra“ ir eilutėje „*is*“ didžiausias skaičius bus stulpelio „yra“ langelyje. Palaipsniui ir visi kiti žodžiai yra

priskiriami teisingai. Taigi, pavyzdžiui, tikimybė, kad angliškas žodis „*powerful*“ bus išverstas kaip „galingas“ yra 3/10 = 0.3 arba 30 %.

Po to, kai *EM* algoritmas atliko savo darbą, lygiagretus tekstynas yra išdėstomas abiem kryptimis (pavyzdžiui, lietuvių-anglų ir anglų-lietuvių). Gaunami du žodžių priskyrimo atvejai, kurie turi būti suvienyti (2 pav.).



2 pav. Anglų-lietuvių ir lietuvių-anglų tekstyno du žodžių priskyrimo atvejai, žemiau pateikiamas šių išdėstymų suvienijimo (simetrizavimo) pavyzdys: tamsesni langeliai – sankirta, šviesesni – sąjunga

Po to mokslininkai naudoja skirtingus metodus, tačiau populiariausias yra Ocho ir Ney būdas (*GIZA++* metodo patobulinimas), pagrįstas *IBM* ketvirtuoju modeliu sudarytais žodiniais priskyrimais. Autoriai iš pradžių sudaro simetrizuotą priskyrimo variantą imdami abiejų priskyrimų aibių sankirtą, o toliau prideda kitus ryšius remdamiesi įvairiomis euristikomis.

Frazėmis paremtas statistinis vertimas dominuoja. Nepaisant to galima teigti, kad populiarumo sulaukė statistinių vertimo sistemų papildymo sintaksine ar morfologine informacija metodas, o taip pat ir sistemų integracijos būdas.

1.2.4. Lingvistinės informacijos turinčios statistinės vertimo sistemos

Statistinės vertimo sistemos, turinčios lingvistinės informacijos, populiarėja dėl galimybės gauti taisyklingą ir adekvatų vertimą. Tikslinės kalbos lingvistinės informacijos panaudojimas leidžia pataisyti tokius netikslumus kaip linksniai, prielinksniai, žodžių tvarkos ypatumai, kurie gali būti tik aproksimuojami naudojant n-gramas. T. Albrektas, prisidėjęs prie lietuviškos bendrovės „*Tilde IT*“ automatinio vertimo sistemos kūrimo, teigia, kad lingvistinės informacijos integravimas į vertimo modelį yra naudingas dėl dviejų priežasčių (Albrektas 2010):

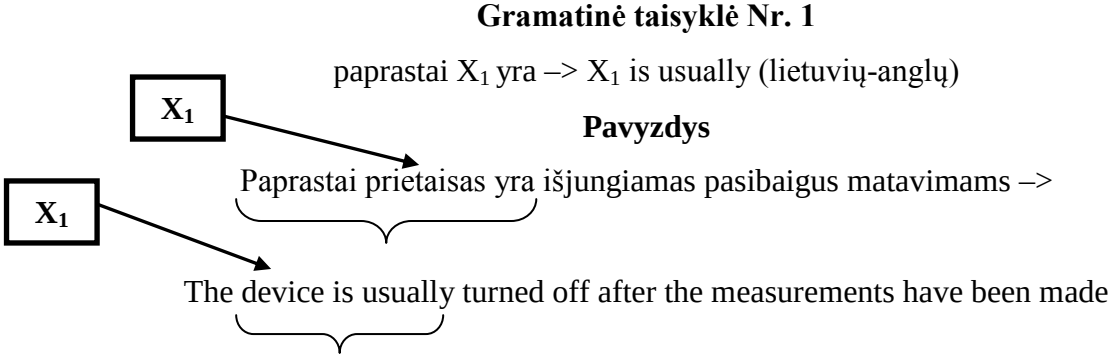
1. Vertimo modeliai, kurie veikia pagal bendresnius atvaizdavimus, pavyzdžiui, lemas (pagrindines žodžių formas), o ne išorines žodžių formas, gali naudotis gausesniais statistiniais duomenimis ir taip įveikti duomenų skurdumo problemas, kurias sukelia riboti mokymo duomenys.

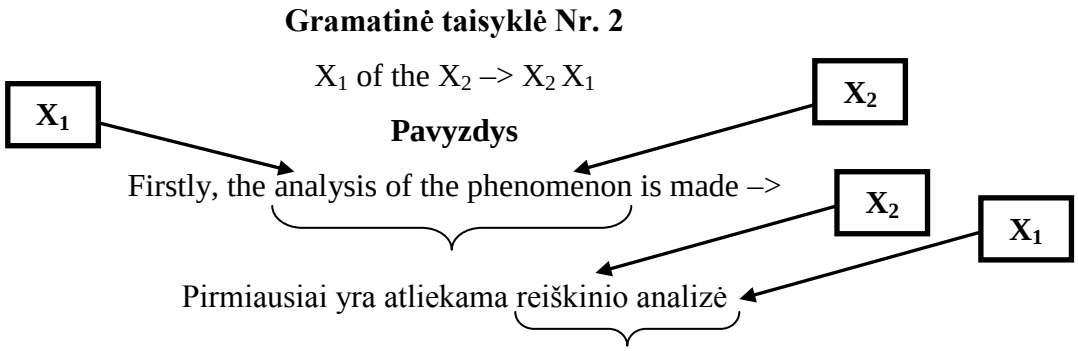
2. Daugelis vertimo aspektų geriausiai gali būti paaiškinami morfologiniu, sintaksiniu ar semantiniu lygmeniu. Turint tokią informaciją, šiuos aspektus galima tiesiogiai modeliuoti vertimo modelyje. Pavyzdžiui, pertvarkymas sakinio lygmeniu dažniausiai atliekamas pagal bendruosius sintaksės principus, vietiniai žodžių derinimo suvaržymai išryškėja, kai atsižvelgiama į morfologiją ir t.t.

Egzistuoja nemažai būdų kaip panaudoti kalbos ypatumus statistinėse vertimo sistemose. Du populiarūs metodai yra pateikiami žemiau.

1.2.4.1. Modeliai paremti sintaksiniais medžiais

Vienas iš jų yra medžiais paremti modeliai (angl. *tree-based models*). Tradiciniai fraziniai vertimo modeliai vykdo taisykles, kuriuose galimas tik priskyrimas tarp įvestos frazės ir išvestos frazės. Medžiais paremti modeliai operuoja remiantis gramatinėmis taisyklėmis, kas leidžia naudoti kintamuosius priskyrimo taisyklėse:



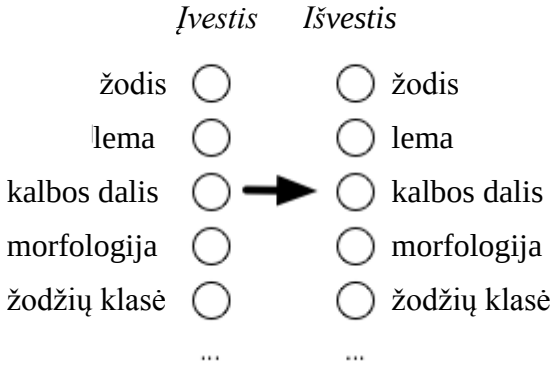


3 pav. Sintaksinės vertimo sistemos gramatinių taisyklių pavyzdys

Kintamieji X_1 ir X_2 šiose gramatinėse taisyklėse gali įgyti ne tik *X* pavidalą, tačiau ir, pavyzdžiui, *NP* (angl. *noun phrase* – daiktavardinė frazė) arba *VP* (angl. *verb phrase* – veiksmažodinė frazė) pavidalą. Šis metodas vadinamas medžiais paremtu, kadangi vertimo metu konstruojama medžio struktūra.

1.2.4.2. Faktorizuoti modeliai

Dauguma mokslinių straipsnių automatinio vertimo tematika vartoja anglų kalbą kaip tikslinę. Verčiant į kitas kalbas kyla problemų, kurios yra nepalyginamai mažesnės negu anglų kalbos. Morfologine prasme anglų kalba yra žymiai paprastesnė nei kitos kalbos, kur veiksmažodžiai ir daiktavardžiai gali turėti daug įvairių formų. Tokių kalbų žodynai yra žymiai didesni ir yra sunkiau surinkti patikimą statistiką. Pavyzdžiui, versti iš anglų į morfologiškai turtingą rusų kalbą būtų problemiška, kadangi morfologija reikalauja sudėtingos šaltinio analizės, o n-gramų metodas nesugeba įveikti šio uždavinio. Mokslininkai Koehnas ir Hoangas 2007 metais pasiūlė faktorizuotus vertimo metodus, kur šaltinio kalbos žodžiai turi daugiau lingvistinės informacijos (lemos, kalbos dalys, morfologinės žymės). Šiame metode žodis – tai ne vien simbolis, o faktorių vektorius, kuris turi įvairaus lygio anotacijas arba ryšius (4 pav.).



4 pav. Faktorizuotas vertimo modelis, kur žodis susietas su morfologine ir kitokia informacija

1.3. Automatinio vertimo sistemų veikimo principų palyginimas

Taisyklėmis paremtos vertimo sistemos privalumai:

- siūlo pakankamai gerą vertimą ir yra labiau nuspėjama;
- suteikia galimybę naudotojams tobulinti žodynus, todėl garantuoja geresnę kokybę tais atvejais, kai sistema susiduria su naujais terminais;
- nereikalauja galingos kompiuterinės įrangos.

Taisyklėmis paremtos vertimo sistemos trūkumai:

- gautam vertimui gali trūkti sklandumo (angl. *fluency*), pavyzdžiui, bus žodžių tvarkos netikslumų;
- sistema nesusitvarko su turimų taisyklių išimtimis, nebent jos yra iš anksto įvedamos į žodyną.

Statistinių vertimo sistemų privalumai:

- gaunamas vertimas yra sklandus, t.y. jį lengva skaityti ir suprasti;
- žymiai geriau susitvarko su kalbos taisyklių išimtimis, nes turint didelę ir kokybišką tekstų bazę, yra atsižvelgiama į šias išimtis.

Statistinių vertimo sistemų trūkumai:

- statistinės sistemos nėra nuspėjamos, priklauso nuo turimų tekstų dalykinės srities;
- reikalingos didžiulės ir kokybiškos lygiagrečių tekstų bazės, kuriose saugomi kelių žodžių junginiai (n-gramos) ir jų vertimai;
- šių sistemų statistinio modelio sukūrimui ir palaikymui reikalinga sparti kompiuterinė įranga.

Taigi abu populiariausi metodai, kuriais remiasi automatinio vertimo sistemos, turi ir panašumų ir trūkumų. Šiuo metu dominuoja statistinio vertimo sistemos. Šios sistemos nesprendžia bendros automatinio vertimo problemos, o tik atlieka lygiagrečių tekstų rinkinio analizę, todėl jų kūrimas užima mažiau pastangų nei taisyklių vertimo sistemų kūrimas (kur reikia apibrėžti daugybę taisyklių). Nepaisant to, neseniai atlikti mokslininkų tyrimai (Callison-Burch *et al.* 2009) parodė, kad kai kuriais atvejais taisyklėmis paremtos sistemos vis dar gali pasiūlyti tikslesnį vertimą nei statistinės sistemos. Todėl dabar mokslininkai siekia sujungti šiuos pagrindinius metodus ir pasiūlyti mišrią vertimo sistemą, kuri galėtų žymiai pagerinti automatinio vertimo kokybę. Plačiau mišriosios vertimo sistemos bus aptartos kitame skyriuje.

1.4. Automatinio vertimo sistemos lietuvių ir anglų kalbų porai

Kadangi darbe pagrindinis dėmesys yra skirtas lietuvių ir anglų kalbų porai būtų tikslinga pateikti šių kalbų savybes lingvistikos požiūriu.

Lietuvių kalba yra seniausia iki šiol tebenaudojama indoeuropiečių šeimos kalba. Tai yra sudėtinga fleksinė kalba, t.y. tokia kalba, kurioje sintaksiniai santykiai daugiausiai išreiškiami kaitant žodžių galūnes. Lietuvių kalboje yra septyni linksniai, o tai reiškia, kad būdvardžiai ir daiktavardžiai turi iki keturiolikos skirtingų galūnių (vienaskaitos ir daugiskaitos). Be to, būdvardžiai dar yra kaitomi giminėmis (vyriška, moteriška ir bevarde) ir turi vieną įvardžiuotinę ir penkias laipsniavimo formas.

Iš visų indoeuropiečių šeimos kalbų lietuvių kalba išsiskiria didžiausiu veiksmažodinių formų kiekiu. Lietuvių kalbos veiksmažodžiai kaitomi tiek asmeniu, tiek skaičiumi. Be įprastų formų jie taip pat turi liepiamosios ir tariamosios nuosakų formas. Be to, lietuvių kalboje yra kelių rūšių dalyviai, kurie kaip ir veiksmažodžiai turi įvairias laiko formas, o taip pat ir dvi specialios dalyvinės kalbos dalys – pusdalyvis ir padalyvis.

Kalbos vartojimo požiūriu fleksinė lietuvių kalbos prigimtis reiškia, kad žodžių tvarka nėra tokia svarbi kaip izoliacinėse kalbose, tokiose kaip anglų kalba. Todėl anglišką frazę „*computer is turning off*“ galima išversti kaip „kompiuteris išsijungia“ (akcentuojamas veiksnys) arba „išsijungia kompiuteris“ (akcentuojamas tarinys).

Anglų kalba yra pasaulinė mokslo kalba, todėl į daugelį kitų kalbų ateina terminai, kuriuos sugalvojo anglakalbiai mokslininkai. Daug angliškų skolinių prigijo ir lietuvių kalboje, kurioje pastebimas mokslinių sričių terminų trūkumas. Tačiau vis dėlto per pastarąjį dešimtmetį atsirado nemažai lietuviškų svetimų terminų ekvivalentų, ypač tokiose srityse kaip informatika. Įvairių sričių specialistams yra ypač naudinga lietuviškų techninių terminų elektroninė bazė (Lietuvos Respublikos terminų bankas).

Palyginus su kitomis indoeuropiečių šeimos kalbomis anglų kalbos gramatika pasižymi minimalia fleksija. Pavyzdžiui, šiuolaikinė anglų kalba, skirtingai nuo šiuolaikinės vokiečių arba romanų kalbų, neturi būdvardžių giminės. Linksniai anglų kalboje yra beveik išnykę ir linksniuojami yra tik įvardžiai. Šios kalbos veiksmažodžiai nėra nei asmenuojami, nei kaitomi skaičiumi, išskyrus kelis atvejus: veiksmažodį “to be” ir pridedamą prie esamojo laiko vienaskaitos trečiojo asmens galūnę.

Verčiant iš turtinga morfologija pasižyminčios lietuvių kalbos į anglų kalba susiduriama su negausių duomenų (angl. *sparse data*) problema, kuri yra viena iš pagrindinių problemų natūralios kalbos apdorojimo srityje (Peng 2001). Šiuo atveju, jeigu tekstyne, kuriuo paremta vertimo sistema, aptinkamos tik dvi konkretaus žodžio formos (pavyzdžiui, vardininko ir galininko linksnių formos), tai pateikę vertimo sistemai sakinį su kita šio žodžio forma (pavyzdžiui, naudininko linksnio forma) žodis išverstas nebus. Tačiau ir kitu atveju, kai kalbos yra sukeistos vietomis, negausių duomenų problema išlieka. Pavyzdžiui, angliškas žodis „*system*“ gali turėti keturis lietuviškus atitikmenis – „sistema“, „sistemos“,

„sistemą“ ir „sistemoje“. Jis bus išverstas kaip vienas iš šių lietuviškų variantų, bet dėl duomenų stokos sistema negali tiksliai nustatyti teisingos žodžio formos.

Taigi anglų kalba yra žymiai labiau analitinė negu lietuvių, nes prasmei perduoti naudoja pagalbinius veiksmažodžius ir griežtą žodžių tvarką, o ne įvairius žodžių darinius.

1.4.1. Automatinio vertimo sistema *Google Translate*

Bene populiariausia šiuolaikinė automatinio vertimo sistema yra statistinė vertimo sistema *Google Translate*, kuri atsirado 2001 m. Pirmoji sistemos versija buvo skirta tik tinklalapiams versti ir palaikė penkias kalbas. Tais pačiais metais atsirado galimybė išversti bet kokį įvestą tekstą. Pažymėtina tai, kad *Google* automatinis vertėjas iš pradžių naudojo taisyklių metodą ir buvo paremtas *SYSTRAN* vertimo sistema. Tačiau 2007 m. pagrindinis *Google* kompanijos vertimo srities tyrėjas Francas Ochas nusprendė atsisakyti šio metodo ir pereiti prie savo sukurtos statistinės vertimo metodikos. Dabartinė *Google Translate* vertimo sistemos grafinė naudotojo sąsajos versija (*Google* vertėjas) yra pavaizduota žemiau:

5 pav. Automatinio vertimo sistemos *Google Translate* grafinė naudotojo sąsaja: 1 – kalbos, iš kurios yra verčiamas tekstas, pasirinkimas (iš viso yra 64 kalbos), 2 – sukeisti vertimo kalbas vietomis, 3 – kalbos, į kurią yra verčiamas tekstas, pasirinkimas, 4 – vertimo mygtukas, 5 – klaviatūra, kuri leidžia įvesti pasirinktai kalbai būdingas raides (pavyzdžiui, lietuvių kalbos raidės “ą”, “ž” ir t.t.), 6 – langas tekstui, kurį norima išversti, 7 – langas išverstam tekstui, 8 – meniu, atsirandantis paspaudus ant konkrečios n-gramos ir leidžiantis pažiūrėti kitus galimus vertimo variantus, 9 – išversto teksto įgarsinimas, 10 – funkcija, leidžianti naudotojui įvertinti pateiktą vertimą

Sprendžiant iš 5 pav., *Google Translate* grafinė naudotojo sąsaja yra tikrai funkcionali. Ypač naudinga yra galimybė įvesti tam tikrai kalbai būdingas raides ir galimybė pasiklausyti, kaip skamba išverstas tekstas. Be to, *Google Translate* atsižvelgia į naudotojų nuomonę. Naudotojas gali pakeisti žodžių tvarka sakinyje ir pasiūlyti ją kaip teisingą arba pasakyti sistemai, kad tam tikri žodžiai turi būti pakeisti į kitus variantus (jie pateikiami paspaudus pele ant dominančio žodžio ar frazės). Taip pat

kiekvienas naudotojas gali įkelti savo turimas vertimo atmintis (bazė iš teksto segmentų, saugomų dviem kalbom) į specialų vertėjo įrankių komplektą ir taip pagerinti vertimo kokybę.

Google Translate vertimo sistema yra paremta milžiniškais tekstų kiekiais. Nemažą dalį šių tekstų sudaro Jungtinių Tautų Organizacijos dokumentai, kurie yra publikuojami šešiomis oficialiomis šios organizacijos kalbomis. Dideli turimų duomenų kiekiai leido *Google* kompanijai sudaryti septintos eilės n-gramų kalbos modelį anglų kalbai. Tokį modelį sudaro daugiau negu vienas trilijonas žodžių (Norvig 2007). Žodžiais šiuo atveju vadinamos visos įmanomos žodžių formos, asmenvardžiai, geografiniai pavadinimai ir kt.

Google Translate turi aplikacijos programavimo sąsają (angl. *application programming interface – API*), kuri leidžia programų kūrėjams integruoti šią vertimo paslaugą į savo programas. 2011 metų gruodžio 1 dieną *Google Translate API* oficialiai tapo mokama paslauga. Tačiau tokių programų kaip *Google Translator* (Bhavnani 2010) egzistavimas parodo, kad *Google Translate API* vis dar galima naudoti nemokamai. *Google* kompanijos vertimo sistema iki šiol neidentifikuoja naudotojų ir tebenaudoja labai paprasto pavidalo užklausą *HTTP* protokolu:

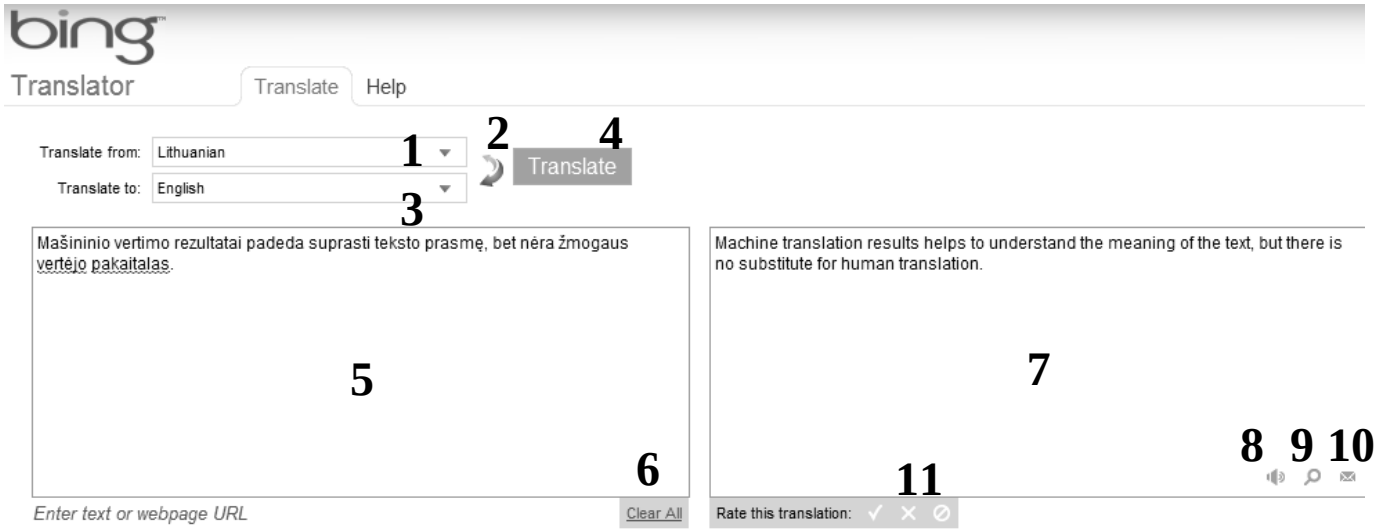
http://www.google.com/translate_t?text=nemokamas vertimas&langpair=lt|en

http://translate.google.com/?hl=en&ie=UTF8&text=free translation&langpair=en|lt

Galima naudoti bet kurią iš dviejų aukščiau pateiktų užklausų. Pirmojoje užklausose nurodoma funkcija *translate_t*, kurios argumentais gali būti *hl* – vertimo tinklalapio kalba, *ie* – naršyklės koduotė, *text* – tekstas, kurį reikia išversti ir *langpair* – vertimo kalbų pora (iš pradžių nurodomas pradinės kalbos trumpinys, po to – kalbos, į kurią verčiamas tekstas, trumpinys). Tokie patys argumentai gali būti nurodyti ir antrojoje užklausoje.

1.4.2. Automatinio vertimo sistema *Microsoft Bing Translator*

Kompanijos *Microsoft* atsakas *Google Translate* vertimo sistemai buvo sistema *Bing Translator*, kuri pradėjo veikti 2009 m. Ši vertimo sistema, kaip ir jos pagrindinė konkurentė, yra paremta statistiniu vertimo principu. *Bing Translator* palaiko 38 pasaulio kalbas, leidžia versti tinklalapius ir įgarsinti išverstą tekstą. Dabartinė vertimo sistemos grafinė naudotojo sąsajos versija (*Bing Translator*) atrodo taip:



6 pav. Automatinio vertimo sistemos *Bing Translator* grafinė naudotojo sąsaja: 1 – kalbos, iš kurios yra verčiamas tekstas, pasirinkimas (38 kalbos), 2 – sukeisti vertimo kalbas vietomis, 3 – kalbos, į kurią yra verčiamas tekstas, pasirinkimas, 4 – vertimo mygtukas, 5 – langas tekstui, kurį norima išversti, 6 – pašalinti tekstą iš įvedimo lango, 7 – langas išverstam tekstui, 8 – išversto teksto įgarsinimas, 9 – ieškoti pateikto vertimo saityne, 10 – išsiųsti vertimą elektroniniu paštu, 11 – funkcija, leidžianti naudotojui įvertinti pateiktą vertimą

Sprendžiant iš 6 pav., *Bing Translator* grafinė naudotojo sąsaja savo funkcionalumu nusileidžia *Google Translate* vertimo sistemai. Sąsaja yra pateikiama vienintele kalba – anglų. Taip pat nėra pagalbinės klaviatūros funkcijos tiems naudotojams, kurie įveda tekstą unikalių raidžių turinčia kalba. Funkcija, leidžianti ieškoti pateikto vertimo saityne, yra naudinga tik verčiant frazes ar labai trumpus sakinius, nes akivaizdu, kad paprasto ilgio sakinys ar pastraipa nebus surasti. Galiausiai gautas vertimas gali būti nusiųstas elektroniniu paštu, tačiau siuntėjo kompiuteryje turi būti įdiegta *Microsoft* pašto kliento programa, tokią kaip, pavyzdžiui, *Microsoft Outlook*.

Kalbant apie naudotojo dalyvavimą tobulinant sistemą, *Bing Translator* kol kas siūlo tik standartinius vertimo įvertinimo mygtukus. Taip pat sistemos tinklalapyje yra speciali skiltis, vadinama *Language labs*, kur naudotojas gali pažiūrėti į *Bing Translator* kalbos modelio siūlomus įvesto sakinio skirtingus užrašymo variantus (tik anglų kalba).

Bing Translator kaip ir *Google Translate* turi aplikacijos programavimo sąsają. Neapribotas *Bing Translator API* yra mokamas. Nemokama versija leidžia išversti iki dviejų milijonų simbolių per mėnesį, tačiau tam būtina užsiregistruoti ir gauti aplikacijos *ID*, be kurio negalima siųsti užklausų. Taigi *Bing Translator* užklausa *HTTP* protokolu turi naudotoją identifikuojančią eilutę:

http://api.microsofttranslator.com/V2/Ajax.svc/Translate?appId=F1B50AB0743B541AA8C07089042D7B57E9B28D25&from=lt&to=en&text=norėtume gauti geriausią šio sakinio vertimą

Aukščiau pateiktoje užklausoje vertimo funkcija *Translate* turi tokius argumentus: *appId* – unikaliai naudotoją identifikuojanti eilutė, *from* – pradinės kalbos trumpinys, *to* – kalbos, į kurią verčiamas tekstas, trumpinys ir *text* – tekstas.

1.4.3. Automatinio vertimo sistema *Moses*

Formaliai *Moses* nėra vertimo sistema, o tik įrankis, kuriuo, turint lygiagretų tekstyną, galima automatiškai sukurti statistinę vertimo sistemą bet kuriai kalbų porai. Tačiau patogumo dėlei *Moses* ir toliau bus vadinamas vertimo sistema. *Moses* yra atvirojo kodo ir buvo sukurta remiantis *Pharaoh* sistema, kuri, savo ruožtu, buvo realizuota vieno iš aktyviausių statistinio vertimo srities mokslininkų Philippo Koehno pastangomis *IBM* modelių pagrindu. *Moses* pagrindinės savybės:

- palaikomi du vertimo modelių tipai: paremtas frazėmis ir paremtas medžiais tipai;
- palaikomas faktorizuotas vertimo modelis, kuris leidžia žodiniams vienetams priskirti lingvistinę ir kitokią informaciją;
- palaikomi sumaišymo tinklai ir gardelės, kas leidžia integruoti šią vertimo sistemą su netikslius įvedimo duomenis generuojančiais įrankiais;
- spindulio paieška (angl. *beam search*) – efektyvus paieškos algoritmas, greitai randantis labiausiai tikėtiną vertimą didėjančiame variantų skaičiuje.

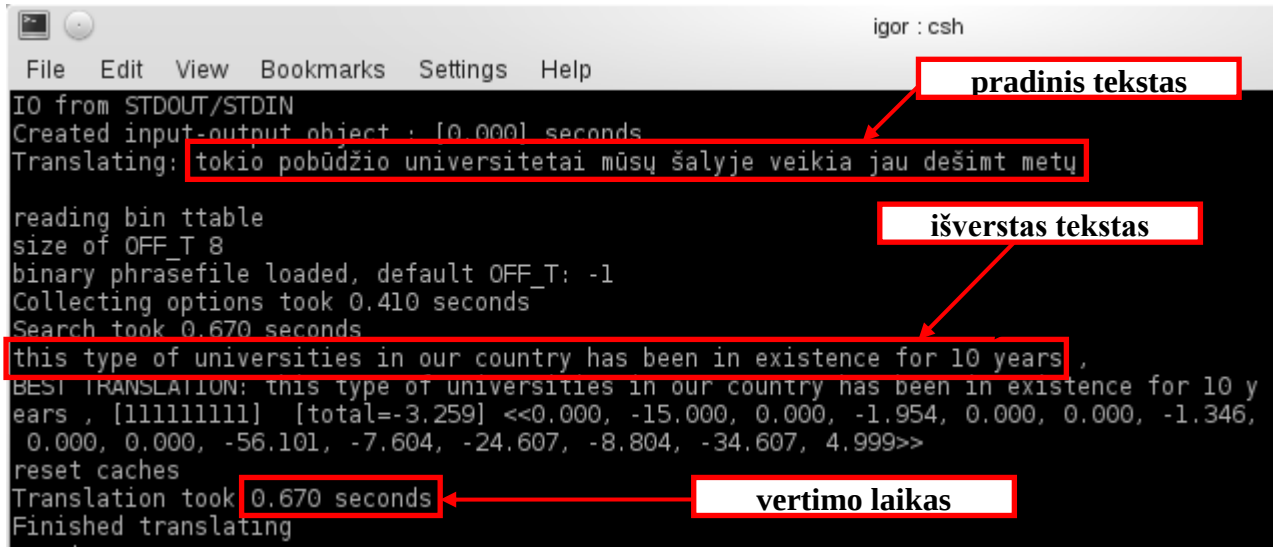
Moses įrankis turi komponentus, reikalingus duomenims apdoroti, kalbos ir vertimo modeliams sukurti, šiems modeliams sukonfigūruoti naudojant minimalaus klaidų dažnumo apmokymą ir gautam vertimui įvertinti remiantis *BLEU* metrika. *Moses* taip pat naudoja išorinius įrankius *GIZA++* žodžių priskyrimui ir *SRILM* kalbos modelių sukūrimui. Be to, kadangi šios užduotys labai užima procesorių, *Moses* veikia lygiagrečioje aplinkoje *Sun Grid Engine*, kad padidintų našumą. *Moses* sistema parašyta *C++* programavimo kalba remiantis objektinio programavimo paradigmomis. Sistemą galima naudoti kaip sukonfigūruoto serverio paslaugą arba dirbant su „debesimi“ (angl. *cloud*) skaičiavimo platformoje *Amazon EC2* (Koehn *et al.* 2007; Moses/Advanced features).

Moses vertimo sistemos sąsaja yra tekstinė (žr. 7 pav.), o ne grafinė, todėl norint išversti tekstą jį reikėtų įvesti per operacinės sistemos komandinę eilutę. Žemiau yra pateikiamas Moses vertimo komandos pavyzdys:

echo "tokio pobūdžio universitetai mūsų šalyje veikia jau dešimt metų" |

/home/igor/moses/moses-cmd/src/moses -f /home/igor/moses/work/tuning/moses-tuned-bin.ini

Aukščiau pateiktame pavyzdyje standartinė *Linux* operacinių sistemų *echo* komanda per konvejerio operatorių paduoda tekstą vykdomajam failui, kuris yra kataloge */home/igor/moses/moses-cmd/src/*. Raktas *-f* nurodo kokį *Moses* vertimo sistemos nustatymų failą naudoti konkrečiu atveju.



7 pav. Automatinio vertimo sistemos *Moses* tekstinė naudotojo sąsaja

Kadangi *Moses* palaiko frazėmis paremtus modelius, frazių lentelė (vertimo modelis) turi ne tik pavienių žodžių, bet ir daugiažodžių įrašų:

sustokite ||| *stop* ||| *0.0104712 0.0036232 1 1 2.718* ||| ||| *382 4*

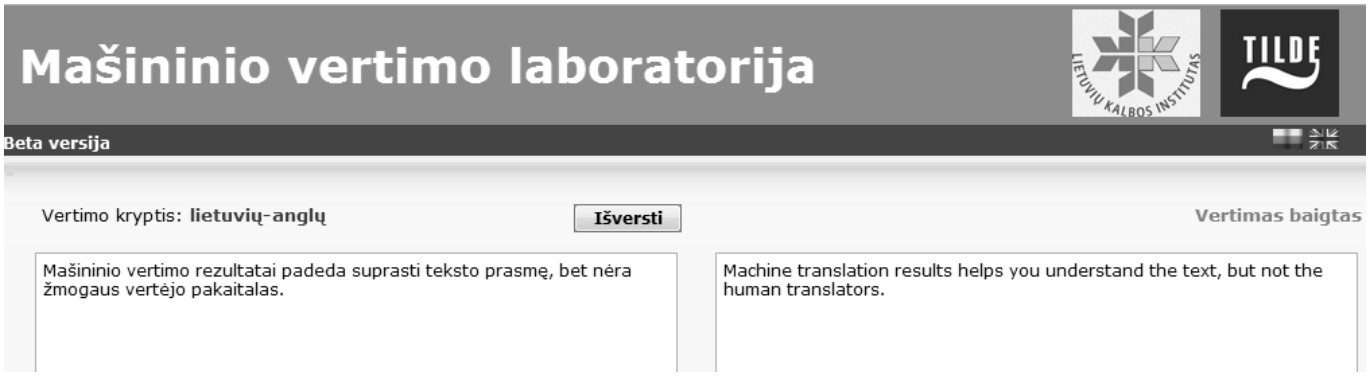
sustokite ir ||| *stop and* ||| *0.166667 0.00290637 1 0.800462 2.718* ||| ||| *12 2*

sustokite ir pagalvokite ||| *stop and think* ||| *0.285714 1.3369e-05 1 0.498649 2.718* ||| ||| *7 2*

Pirmasis ir trečiasis kiekvieno įrašo skaičiai yra svarbiausi ir parodo vertimo tikimybes. Pavyzdžiui, angliško žodžio „*stop*“ vertimo į lietuvišką „*sustokite*“ tikimybė yra lygi *p(sustokite | stop) = 0.0104712* (pirmasis skaičius). Priešingu atveju tikimybė, kad lietuviškas žodis „*sustokite*“ bus išverstas į anglų kalbą kaip žodis „*stop*“, įgyja didžiausią įmanomą reikšmę – vienetą (trečiasis skaičius).

1.4.4. Lietuvių kalbos instituto ir „*Tilde IT*“ automatinio vertimo sistema

Šią eksperimentinę automatinę vertimo sistemą parengė Lietuvių kalbos institutas (*LKI*) kartu su „*Tilde IT*". Bendrovė „*Tilde IT*“ ištyrė galimas vertimo platformas ir nusprendė naudoti anksčiau pristatytą atvirojo kodo statistinio vertimo įrankį *Moses*. Bandomoji sistemos versija buvo paleista 2010 m. Šiuo metu vertimo sistema gali išversti lietuvių kalbos tekstą į anglų kalbą. Dabartinė grafinė naudotojo sąsaja yra minimali (Mašininis vertimas) ir yra pavaizduota žemiau esančiame paveiksle:



8 pav. *LKI* ir *Tildė IT* automatinio vertimo sistemos grafinė naudotojo sąsaja

Pažymėtina tai, kad ši internetinė vertimo sistema yra demonstracinė ir neturi jokių papildomų funkcijų. Žymiai patobulinta šios sistemos versija yra mokama ir 2012 m. buvo pirmą kartą buvo integruota į „*Tilde IT*“ reguliariai išleidžiamą kompiuterinę programą „*Tildės Biuras*“. Vertimo sistema, integruota į paskutinę programos versiją „*Tildės Biuras 2012*“ leidžia versti tekstą tiek iš anglų į lietuvių, tiek iš lietuvių į anglų kalbas. Bendrovės pranešime spaudai (Tilde IT) teigiama, kad „produkte „*Tildės Biuras 2012*“ integruota automatinio vertimo sistema „lenkia ir IT giganto *Google* mašininį vertimą – tai rodo kokybinis palyginimas, atliktas naudojant statistinę įvairių anglų-lietuvių kalbų tekstų analizę pagal *BLEU/NIST* metriką, plačiai taikomą tarptautiniuose mašininio vertimo tyrimuose“. „*Tilde IT*“ specialistai savo automatinę vertimo sistemą sukūrė po kelerius metus trukusių sudėtingų mokslinių tyrimų. Kadangi bendrovė sistemai sukurti panaudojo *Moses* įrankį, buvo nuspręsta pasinaudoti sparta pasižyminčia *Amazon* debesų kompiuterijos paslauga.

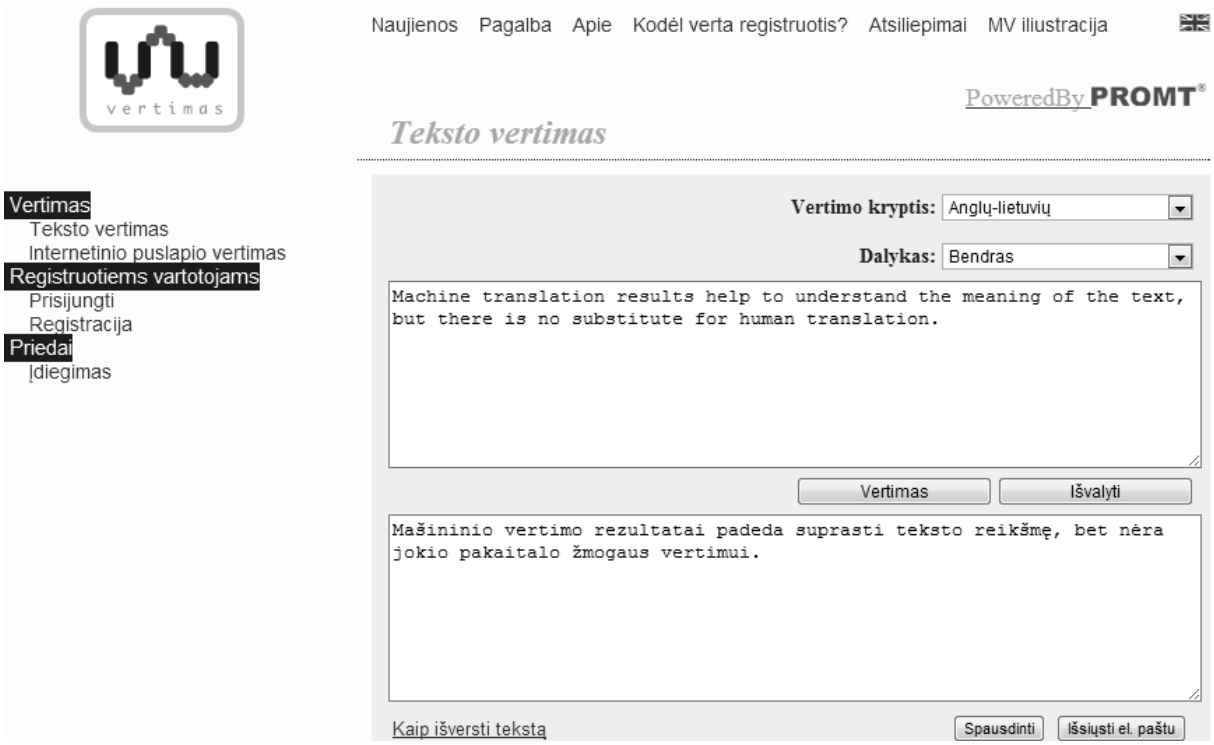
LKI ir „*Tilde IT*“ sukurta vertimo sistema naudoja *ASP.NET* tinklo aplikacijų platformą. Užklausa vertimui gauti *HTTP* protokolu yra gana paprasta:

http://mvlab.lki.lt/getTranslation.aspx?&text=Norėtume gauti geriausią šio sakinio vertimą.

Vertimo darbą atlieka *ASP.NET* forma pavadinimu *getTranslation*, kurios pagrindinis argumentas yra *text* – tekstas, kurį norima išversti.

1.4.5. Vytauto Didžiojo universiteto automatinio vertimo sistema

Vytauto Didžiojo universiteto (*VDU*) automatinio vertimo sistema buvo sukurta 2007 m. projekto “Internetinė informacijos vertimo priemonė” metu. Šis projektas buvo finansuotas Europos Sąjungos struktūrinių fondų. *VDU* vertimo sistema skiriasi nuo sistemų, pristatytų anksčiau, kadangi naudoja taisyklėmis pagrįsto vertimo technologiją. Jos pagrindą sudaro rusiška vertimo sistema *PROMT*. Dabartinė *VDU* vertimo įrankio grafinė naudotojo sąsaja (Teksto vertimas) yra pateikiama žemiau:



9 pav. *VDU* Automatinio vertimo sistemos grafinė naudotojo sąsaja

VDU automatinio vertimo įrankio grafinė naudotojo sąsaja yra gana paprasta. Sąsaja turi laukus tekstui įvesti ir išverstam tekstui pateikti, o taip pat mygtukus tekstui išversti, išvalyti ir spausdinti. Teksto siuntimo elektroninių paštu funkcija yra neveikianti. Vertimo krypties pasirinkti negalima, kadangi sistema verčia tik viena kryptimi – iš anglų į lietuvių kalbą. Vertimo sistemos tinklalapyje buvo paminėti planai, kurie leistų vertimą pagerinti prie jo prijungiant specializuotus teminius žodynus (9 pav., laukas „Dalykas“) ir vertimo atmintį, tačiau šie planai iki šiol liko neįgyvendinti. Be to, tinklalapyje teigiama, kad prisiregistravus galima versti tekstą naudojant įvairius nustatymus: anglų kalbos variantus (britų/amerikiečių), teksto tipą (pvz., dokumentų vertimas), kelių verčiamų žodžių reikšmių rodymą ir kt. Deja, šių vertimo galimybių išbandyti negalima, kadangi naujų naudotojų registracija šiuo metu neveikia.

VDU vertimo sistemą galima naudoti ir neatidarius naršyklės. Šiam tikslui buvo sukurti specialūs įskiepiai, kurie gali būti integruoti į standartines *Microsoft Windows* operacinės sistemos programas.

Taip pat verta paminėti, kad *VDU* kompiuterinės lingvistikos centras yra sukūręs įvairių naudingų įrankių, tokių kaip morfologinis anotatorius (pateikia įvestų žodžių lemas) ir teksto analizatorių, kuris pateikia tekste pasitaikančių žodžių dažnumą ir kitus duomenis (KLC).

1.5. Automatinio vertimo įvertinimas

Kaip galima įvertinti šiuolaikinės statistinio vertimo sistemos vertimo kokybę? Atsakyti į šį klausimą nėra lengva. Palyginus su kitais kalbos uždaviniais, tokiais kaip kalbos atpažinimas, galima

teigti, kad neegzistuoja joks vienas vertimo variantas, kurį vertimo sistema turėtų pateikti. Akivaizdu, kad keli vertėjai, verčiantys tą patį (netgi trumpą) sakinį, pateiks skirtingus vertimo variantus. Egzistuoja du būdai automatinio vertimo kokybei įvertinti. Pirmas būdas – pasamdyti kalbų žinovus, kurie išanalizuos vertimą ir pateiks savo pastabas. Antras būdas – leisti kompiuteriui nustatyti automatinio vertimo panašumą į vertėjo išverstą teksto variantą.

Šiame skyriuje pristatomi skirtingi šiuolaikiniai būdai, kuriuos naudoja automatinio vertimo tyrėjai. Automatinio vertimo įvertinimas tebelieka iki galo neišspręsta problema ir iki šiol sukelia daug diskusijų tarp mokslininkų.

1.5.1. Rankinis automatinio vertimo įvertinimo būdas

Pats paprasčiausias būdas automatiniam vertimui įvertinti yra rankiniu būdu išanalizuoti jo rezultatus ir nuspręsti ar pateiktas vertimas yra patenkinamas. Šį darbą geriausiai galėtų atlikti specialistai, suprantantys abi (pradinę ir galutinę) vertimo kalbas. Kadangi dažnai nėra galimybės pasamdyti tokių specialistų, dažnai automatinio vertimo rezultatus įvertina žmonės, išmanantys tik vieną kalbą – kalbą, į kurią verčiamas pradinis tekstas. Šiuo atveju jie turi mokėti palyginti gautą tekstą su „atramos“ vertimu (angl. *reference translation*), kuris yra laikomas „teisingu“ vertimo variantu.

Vertimą vertinantys specialistai dažniausiai naudoja tam tikrą gradavimo skalę (žr. 2 lentelę). Kadangi vertimo teisingumas yra per plati sąvoka taip pat naudojami du papildomi kriterijai: kalbos sklandumas (angl. *fluency*) ir adekvatumas (angl. *adequacy*). Tarkime, kad yra verčiama iš anglų kalbos į lietuvių kalbą. Tuomet vertindami sklandumą svarstytume ar gautas vertimas pasižymi sklandžia lietuviška kalba. Kalbos sklandumas – tai ne tik gramatinis taisyklingumas, bet ir panaudoti tinkami pagal sakinio prasmę žodžiai. Įvertinus sklandumą reikėtų įvertinti ir kalbos adekvatumą. Adekvatus vertimas turėtų perduoti tą pačią prasmę, kokią turėjo originalus tekstas. Pamesti arba iškraipyti esminiai žodžiai yra neadekvataus vertimo požymiai.

2 lentelė. Automatinio vertimo teisingumo kriterijų (sklandumo ir adekvatumo) gradavimo skalė

Vertimo kalbos adekvatumas		Vertimo kalbos sklandumas	
5	pradinio teksto prasmė visiškai perduota	5	nepriekaištinga kalba
4	didžioji dalis prasmės perduota	4	gera kalba
3	pakankama dalis prasmės perduota	3	kalba, turinti netikslumų (primenanti žmonių, kuriems ši kalba nėra gimtoji, kalbą)
2	maža dalis prasmės perduota	2	nesklandi kalba
1	prasmė visiškai neperduota	1	visiškai nesuprantama kalba

Lentelėje pateikti apibrėžimai nėra tikslūs, todėl įvertinimams gali trūkti nuoseklumo. Taip pat kai kurie specialistai gali būti pernelyg atlaidūs (pavyzdžiui, jų įvertinimų vidurkis sieks apie 4 balus), kiti – griežti (pavyzdžiui, jų įvertinimų vidurkis bus apie 2 balus).

Metrika, tinkanti automatinio vertimo įvertinimui atlikti, turi pasižymėti ***mažomis sąnaudomis***. Tai reiškia, kad naudojant tokią metriką naujos vertimo sistemos įvertinimas turi būti pakankamai greitas ir pigus. Akivaizdu, kad taikant rankinį automatinio vertimo įvertinimo būdą metrikos sąnaudos yra didelės, nes įvertinimą atlieka žmonės. Sąnaudos gali būti išreikštos laiko arba piniginiais vienetais, kurie buvo išnaudoti atliekant įvertinimą.

Automatinio vertimo įvertinimo metrika taip pat turi pasižymėti ***nuoseklumu***. Skirtingi įvertintojai, naudojantys tą pačią metriką, turi prieiti prie tų pačių išvadų. Be to, vertimą vertinantys specialistai turėtų turėti vieningą nuomonę ne tik apie vieną tekstyno dalį, bet ir apie kitas to paties tekstyno dalis. Jeigu įvertintojų nuomonės žymiai nesutampa, metrika nėra stabili ir tam, kad būtų pasiekti patikimi rezultatai, reikės atlikti labai didelio tekstyno analizę.

Galiausiai automatinio vertimo įvertinimo metrika turi pateikti ***teisingus*** rezultatus. Tačiau, deja, kaip rodo praktika, galima tik palyginti rezultatus, gautus naudojant skirtingas metrikas, bet negalima užtikrintai teigti kad konkreti metrika padeda teisingai įvertinti vertimą. Kadangi kalbos sklandumas ir adekvatumas išlieka dviem patikimiausiais kriterijais būtent jais yra paremtos visos naujos metrikos. Viena iš tokių metrikų yra *BLEU,* kuri bus pristatyta vėlesniame poskyryje.

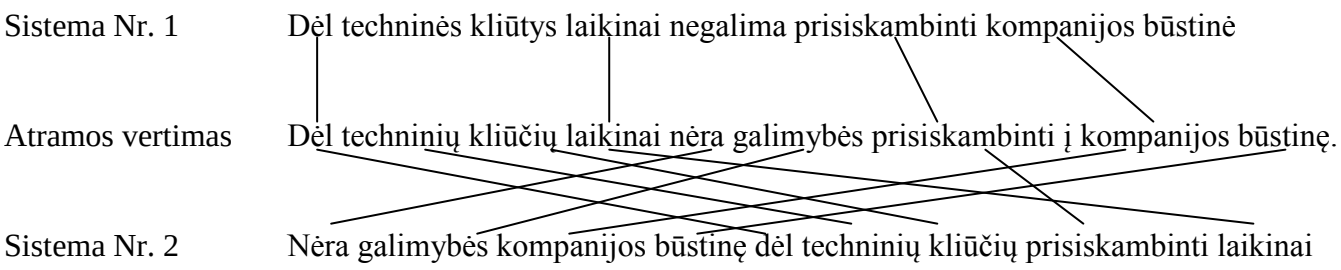
1.5.2. Automatinis vertimo įvertinimo būdas

Kalbant apie automatinio vertimo sistemų įvertinimą daugiausiai yra pasitikima žmonėmis-vertintojais, kurie pažvelgia į kelių vertimo sistemų rezultatus, išanalizuoja ir įvertina kiekvieną gautą sakinį ir galiausiai pateikia galutinį sistemos įvertinimą. Tačiau šis metodas turi didelį trūkumą – dideles sąnaudas. Specialistai, vertinantys vertimą, dirba pakankamai lėtai ir, be to, jiems reikia mokėti ženklų atlyginimą. Paprasti statistinio automatinio vertimo tyrėjai neturi pakankamų piniginių lėšų ir norėtų atlikti daugkartinius vertimo įvertinimus (pavyzdžiui, įvertinti kelias skirtingas sistemos konfigūracijas) per trumpą laiko tarpą. Todėl naudotinas automatinis vertimo įvertinimo metodas. Geriausiu atveju yra norima gauti greitą kompiuterinės programos atsakymą į klausimą: „Ar paskutinė vertimo sistemos versija pateikė geresnį vertimą nei ankstesnioji šios sistemos versija?“

Pastaraisiais metais automatinis vertimo įvertinimas smarkiai pažengė į priekį. Automatinio vertimo tyrėjai taip pasitiki automatinio įvertinimo metrikų balais, kad jais remdamiesi kuria ir keičia

savo vertimo sistemas. Tačiau automatinio įvertinimo metrikos vis dar sukelia nemažai diskusijų ir kai kurie mokslininkai stengiasi paneigti jų naudą vertinant geras ir blogas vertimo sistemas.

Kaip kompiuteris gali įvertinti vertimo kokybę? Visos automatinio įvertinimo metrikos remiasi vienu ir tuo pačiu principu. Kiekvienos sistemos išverstas sakinys yra palyginamas su vienu ar daugiau profesionalių vertėjų pateiktų to paties sakinio variantų (atramos vertimas). Akivaizdu, kad automatinės sistemos arba žmogaus-vertėjo pateikti vertimai negali visiškai sutapti su atramos vertimu. Tačiau visgi labiau tikėtina, kad vertimas, kuris yra panašus į atramos vertimą, yra teisingas. Taigi reikia rasti tokią automatinio vertimo įvertinimo metriką, kuri būtų patikimas šio panašumo matas.



10 pav. Sutampančių žodžių vertimo sistemų ir atramos vertimo sakiniuose paieška: pirmoji sistema pateikė tris teisingus žodžius, o antroji – šešis

Pažvelkime į įvertinimo metriką, kuri pagrįsta sutampančių žodžių paieška (žr. 10 pav.). Tarkime, kad verčiamas sakinys yra: „*The company's headquarters temporarily cannot be reached by phone due to technical difficulties*." Pirmoji sistema pateikė tokį vertimą: „Dėl techninės kliūtys laikinai negalima prisiskambinti kompanijos būstinė". Pateiktame sakinyje yra keturi žodžiai (dėl, laikinai, prisiskambinti, kompanijos), kuriuos galima rasti atramos vertimo variante: „Dėl techninių kliūčių laikinai nėra galimybės prisiskambinti į kompanijos būstinę". Taigi pirmos sistemos pateiktame sakinyje yra 8 žodžiai. Keturi žodžiai buvo išversti teisingai, todėl padalinus šį skaičių iš aštuonių gaunamas 50 % santykis. Gautas santykis yra metrika, kuri informacijos išgavimo teorijoje yra vadinama ***tikslumu*** (angl. *precision*). Toliau panagrinėkime antrosios sistemos vertimą: „Nėra galimybės kompanijos būstinę dėl techninių kliūčių prisiskambinti laikinai". Pateiktas vertimo variantas susideda iš devynių žodžių, kurie visi yra ir atramos vertime. Devyni teisingi žodžiai iš devynių – tai šimtaprocentinis tikslumas. Tačiau akivaizdu, kad antros sistemos pateiktas sakinys nėra gerai išverstas, nepaisant tobulo tikslumo. Pavyzdžiui, žodžiai sakinyje nėra išdėstyti teisingai. Be to, buvo pamestas prielinksnis „į". Taigi patikimai įvertinti vertimo sakinių nepavyks, jeigu dėmesys bus sutelktas vien į žodžių atitiktis, bet ne į jų tvarką. Todėl, reikėtų skaičiuoti kiek teisingų žodžių sistema turi pateikti, o ne kiek žodžių ji iš tikrųjų pateikia. Tokia metrika yra vadinama ***pateikimu*** (angl. *recall*). Pateikimas skiriasi nu tikslumo tuo, kad

teisingai išverstų žodžių sakinyje skaičius yra dalinamas iš atramos vertimo žodžių skaičiaus, o ne iš pateikto sistemos sakinio žodžių skaičiaus:

$$\text{tikslumas} = \frac{\text{teisingai išversti žodžiai}}{\text{sistemos vertimo ilgis}} \tag{7}$$

$$\text{pateikimas} = \frac{\text{teisingai išversti žodžiai}}{\text{atramos vertimo ilgis}} \tag{8}$$

Kiek tikslumo tiek pateikimo metrikos susiduria su viena problema – jos gali būti „apgautos“. Kitaip tariant, galima sukurti vertimo sistemas, kurios išvers tik tuos žodžius, dėl kurių nekyla abejonių. Tokiu atveju pateiktas vertimas bus trumpas ir pasižymės dideliu tikslumu (bet mažu pateikimu). Be abejo, galimos ir tokios vertimo sistemos, kurios pateiks didelį žodžių kiekį ir tokiu būdu yra labai tikėtina, kad dalis šių žodžių sutaps su visais atramos vertimo žodžiais. Šis atvejis yra priešingas prieš tai buvusiam – didelis pateikimas, tačiau mažas tikslumas. Taigi tikslumas ir pateikimas yra dažnai naudojamos metrikos natūralios kalbos apdorojimo srityje ir jų svarba priklauso nuo jų panaudojimo atvejų. Vienas dažniausių šių metrikų naudojimo atvejų yra informacijos paieška saityne. Naudotojo ieškoma informacija gali būti daugelyje tinklalapių. Todėl kiekvienam paieškos varikliui yra svarbu pateikti mažą skaičių gerų rezultatų ir atmesti didelį skaičių rezultatų, kurie nėra įdomūs naudotojui. Taigi šiuo atveju tikslumas yra svarbesnis už pateikimą. Tačiau automatinio vertimo srityje abi metrikos yra svarbios. Neteisingai išversti žodžiai nėra pageidautini, tačiau taip pat nenorima praleisti jokios svarbios informacijos. Dažnai yra naudojamas papildomas matas, vadinamas f-matu (angl. *f-measure*), kuris yra apibrėžiamas kaip tikslumo ir pateikimo metrikų harmoninis vidurkis:

$$\text{f - matas} = \frac{\text{tikslumas} \times \text{pateikimas}}{(\text{tikslumas} \times \text{pateikimas})/2} \tag{9}$$

Pakeitę tikslumą ir pateikimą pagal formules (7,8) galime f-matą užrašyti tokiu būdu:

$$\text{f - matas} = \frac{\text{teisingai išversti žodžiai}}{(\text{sistemos vertimo ilgis} + \text{atramos vertimo ilgis})/2} \tag{10}$$

Dar vienas f-mato variantas, naudojamas automatiniam vertimui įvertinti, yra vadinamas „nuo pozicijos nepriklausomu klaidų dažniu“ (angl. *position-independent error rate – PER*). Šis matas panašus į pateikimo metriką tuo, kad turi tokį patį daliklį – atramos vertimo ilgį. Kadangi *PER* yra klaidų dažnis, juo yra matuojami neatitiktys. Ši metrika taip pat išsprendžia pernelyg ilgų vertimo variantų problemą, kadangi atsižvelgia į nereikalingus trintinus žodžius:

$$PER = 1 - \frac{\text{teisingai išverstižodžiai} - \max(0, \text{sistemos vertimo ilgis} - \text{atramos vertimo ilgis})}{\text{atramos vertimo ilgis}} \quad (11)$$

Kitame poskyryje yra pristatoma populiariausia automatinio vertimo įvertinimo metrika BLEU, veikianti panašiai kaip ir PER.

1.5.2.1. Automatinio vertimo įvertinimo metrika *BLEU*

Automatinio vertimo įvertinimo metrika *BLEU* (angl. Bilingual Evaluation Understudy, lietuviškai galima išversti kaip „dvikalbio įvertintojo pakaitalas“) yra populiariausia metrika, kuri veikia panašiai kaip ir PER, o be to ganėtinai gerai įvertina žodžių tvarką sakiniuose. *BLEU* analizuoja ne pavienių žodžių, o didesnių žodžių junginių (n-gramų) atitikimą atramos vertimo n-gramoms.

Pavyzdys, kuris buvo parodytas prieš tai buvusiame poskyryje ir dabar rodantis sakinių n-gramų atitikimą, yra pavaizduotas 11 pav. Keturios pirmosios sistemos vertimo pirmos eilės n-gramos sutapo su atramos vertime esančiomis n-gramomis. Visi antrosios sistemos žodžiai sutampa su atramos vertimo žodžiais: abiejuose sakiniuose yra ketvirtos eilės n-grama „dėl techninių kliūčių laikinai“, antros eilės n-gramos „nėra galimybės“, „kompanijos būstinę“ ir pirmos eilės n-grama „prisiskambinti“.

Sistema Nr. 1	Dėl	techninės kliūtys	laikinai	negalima	prisiskambinti	kompanijos	būstinė
	1 eilės n-grama		1 eilės n-grama		1 eilės n-grama	1 eilės n-grama	
Atramos vertimas	Dėl techninių kliūčių laikinai nėra galimybės prisiskambinti į kompanijos būstinę.						

Sistema Nr. 2	Nėra galimybės	kompanijos būstinę	dėl techninių kliūčių laikinai	prisiskambinti
	2 eilės n-grama	2 eilės n-grama	4 eilės n-grama	1 eilės n-grama

11 pav. Sutampančių n-gramų vertimo sistemų ir atramos vertimo sakiniuose paieška

Turint n-gramų atitikimo duomenis, galima suskaičiuoti n-gramų tikslumą, t.y. tam tikros eilės teisingų n-gramų ir visų šios eilės sistemos pateiktų n-gramų santykis:

Sistema Nr. 1: 1 eilės n-gramų tikslumas yra 4/8 = 0.5, 2 eilės n-gramų tikslumas yra 0/7 = 0, 3 eilės n-gramų tikslumas yra 0/6 = 0, 4 eilės n-gramų tikslumas yra 0/5 = 0.

Sistema Nr. 2: 1 eilės n-gramų tikslumas yra 9/9 = 1, 2 eilės n-gramų tikslumas yra 5/8 = 0.625, 3 eilės n-gramų tikslumas yra 2/7 = 0.286, 4 eilės n-gramų tikslumas yra 1/5 = 0.2.

Dažna tikslumu paremtų metrikų problema yra tai, kad nėra jokių baudos taškų sakiniams, kurie pateikė mažą dalį pradinės informacijos ir yra trumpi. Išskirtinis *BLEU* metrikos požymis yra baudos koeficientas, kuris sumažina per trumpo sakinio įvertinimą. Daugeliu atveju ieškomos sutampančios n-gramos nuo pirmos iki ketvirtos eilės todėl paprastai *BLEU* metrika yra apibrėžiama taip:

$$\text{BLEU - 4} = \min(1, \frac{\text{sistemos vertimo ilgis}}{\text{atramos vertimo ilgis}}) \prod_{i=1}^{4} \text{tikslumas}_i ; \qquad (12)$$

čia min(1, sistemos vertimo ilgis / atramos vertimo ilgis) yra baudos koeficientas.

Taigi *BLEU* metrikos, įvertinančios ketvirtos ir žemesnės eilės n-gramas sakiniuose, reikšmė yra atitinkamų minimumo funkcijos reikšmių ir tikslumo reikšmių sandauga. Akivaizdu, kad *BLEU* įvertinimas yra lygus nuliui, jeigu bent vienos eilės n-gramų tikslumas yra lygus nuliui. Kadangi n-gramų (ypač ketvirtos eilės) tikslumo reikšmės dažnai yra nulinės, *BLEU* metrikos reikšmės skaičiuojamos dideliam sakinių rinkiniui.

Dar viena naujovė, kuria pasižymi *BLEU* metrika yra daugelio atramos vertimų naudojimas. Suprantama, kad vertimui būdinga naudojamų žodžių ir jų tvarkos įvairovė, todėl sunku tikėtis, kad automatinio vertimo sistema pateiks vertimą, sutampantį su vieninteliu turimu atramos vertimu. Kai turime skirtingų vertėjų pateiktus vertimus yra labiau tikėtina, kad nevienareikšmiai pradinio teksto fragmentai bus išversti teisingai.

Tarkime, kad pavyzdys, pateiktas 10 pav. yra papildomas dar vienu atramos vertimu: „Į kompanijos būstinę laikinai negalima prisiskambinti dėl techninių kliūčių“. Tuomet galima įvertinti *BLEU* metriką remiantis dviem atramos vertimais (žr. 3 lentelę). Pavyzdžiui, pirmoji sistema negavo balų už išverstą žodį „negalima“, kuris nėra klaidingas. Kaip matome iš 3 lentelės, šis žodis yra antrajame atramos vertimo variante. Dabar pažvelkime kaip yra skaičiuojama *BLEU* metrika kai turime kelis atramos vertimus. Jeigu sistemos vertimo n-grama egzistuoja bent viename iš turimų atramos vertimo variantų, ji yra laikoma teisinga. Galimas ir toks atvejis, kai n-grama sistemos pateiktame vertime pasikartoja daug kartų (pavyzdžiui, angliškas artikelis „the“, kai verčiama iš lietuvių kalbos į anglų kalbą). Tada visi pasikartojančios n-gramos egzemplioriai laikomi teisingais, jeigu jie tą patį skaičių kartų pasikartoja bent viename iš atramos vertimų. Kitokiu atveju, jeigu, pavyzdžiui, atramos vertimų variantuose konkreti n-grama aptinkama *k* kartų, o sistemos vertime *k+2* kartų, n-grama yra laikoma teisinga tik *k* kartų. Keli skirtingo ilgio atramos vertimai sukelia klausimą: „Kokį atramos vertimo ilgį (žodžių skaičių) derėtų naudoti *BLEU* formulėse?“ Daugelio atramos vertimų atveju yra naudotinas atramos vertimo ilgis, kuris yra artimiausias sistemos vertimo ilgiui. Tarkime, kad vertimo sistema pateikė 10 žodžių ilgio sakinį, o atramos vertimų ilgiai yra 8, 9, 11 ir 12. Tokiu atveju visada pasirenkamas mažesnis atramos vertimo ilgis (devyni).

3 lentelė. Automatinio vertimo įvertinimas naudojant kelis atramos vertimo variantus

Vertimo variantai				
Sistema Nr. 1	Dėl techninės kliūtys laikinai negalima prisiskambinti kompanijos būstinė			
Sistema Nr. 2	Nėra galimybės kompanijos būstinę dėl techninių kliūčių laikinai prisiskambinti			
Atramos vertimas Nr. 1	Dėl techninių kliūčių laikinai nėra galimybės prisiskambinti į kompanijos būstinę.			
Atramos vertimas Nr. 2	Į kompanijos būstinę laikinai negalima prisiskambinti dėl techninių kliūčių.			
Automatinio vertimo variantų įvertinimas				
	Prieinamas tik pirmas atramos vertimas		Prieinami pirmas ir antras atramos vertimai	
Metrika	Sistema Nr. 1	Sistema Nr. 2	Sistema Nr. 1	Sistema Nr. 2
1 eilės n-gramų tikslumas	0.50	1.00	0.63	1.00
2 eilės n-gramų tikslumas	0	0.63	0.29	0.63
3 eilės n-gramų tikslumas	0	0.29	0.17	0.29
4 eilės n-gramų tikslumas	0	0.20	0	0.20
Baudos koeficientas	0.80	0.90	0.89	1.00
BLEU-1	40%	90%	56%	100%
BLEU -2	0%	56%	16%	63%
BLEU -3	0%	16%	3%	18%
BLEU -4	0%	3%	0%	4%

Automatinių vertimo įvertinimo metrikų naudojimas sukelia nemažai diskusijų. Daugumai kompiuterinės lingvistikos mokslininkų kelia abejonių *BLEU* metrikos sugebėjimas teisingai suprasti skirtumus tarp vertimo sistemos vertimo ir atramos vertimų variantų. Be to, *BLEU* yra daugiausiai naudojama statistinio automatinio vertimo tyrėjų. Aukšti statistinės vertimo sistemos *BLEU* balai neskamba įtikinimai kitus vertimo metodus naudojantiems mokslininkams. Galima pateikti tokius pagrindinius teiginius, kritikuojančius *BLEU* metriką:

- *BLEU* metrika neatsižvelgia į įvairių žodžių svarbą. Akivaizdu, kad kai kurie sakinio žodžiai yra svarbesni už kitus. Pavyzdžiui, sistema, kuri nesugebėjo išversti neigimo žodžių, tokių kaip „nėra“

arba „negalima“ smarkiai iškraipo pradinio teksto prasmę. Taigi galima teigti, kad daiktavardžiai, vardai ir veiksmus reiškiantys žodžiai yra svarbesni nei, pavyzdžiui, būdvardžiai ar skyrybos ženklai.

- *BLEU* metrikos veikimo sritis yra labai maža ir ji neįvertina viso sakinio gramatinio sklandumo. Automatinio vertimo sistema gali pateikti vertimą, kurio n-gramos (pavyzdžiui, trijų ar keturių žodžių junginiai) yra gramatiškai teisingos, bet sakiniai vis tiek nebus iki galo sklandūs. Kadangi būtent statistinio vertimo sistemos pasižymi tokiu vertimu, galima teigti, kad kitais vertimo metodais paremtų vertimo sistemų įvertinimas pagal *BLEU* metriką nebus pakankamai objektyvus.
- Balai gauti įvertinus tekstą pagal *BLEU* metriką neturi daug prasmės. Pavyzdžiui, 30% pagal *BLEU* metriką gali reikšti tiek aukštą, tiek žemą sistemos įvertinimą. Taip yra todėl, kad gautiems balams įtakos turi daugelis veiksnių – atramos vertimų skaičius, kalbų pora, vertimui įvertinti skirtų tekstų dalykinė sritis ir kt.
- Atlikti eksperimentai, kurių metu profesionalių vertėjų išversti tekstai buvo įvertinti pagal *BLEU* metriką. Šie vertimai pagal *BLEU* balus labai nežymiai pranoko automatinės sistemos vertimą nors žmogaus vertimas ir turėjo būti įvertintas aukščiau.

Aukščiau išvardyti teiginiai yra teisingi ne tik *BLEU*, bet ir kitoms automatinėms vertimo įvertinimo metrikoms. Tačiau, nepaisant šių stiprių argumentų, yra ir kontrargumentų. Darbo įvade paminėtas *NIST* institutas 2002 m. atliko arabų ir anglų kalbų automatinio vertimo sistemų įvertinimą. Įvertinimą atliko tiek kompiuteris, tiek profesionalus vertimo įvertintojas. Mokslininkai automatiniam įvertinimui panaudojo savo sukurtą metriką *NIST*, kuri yra *BLEU* variantas. Gauti rezultatai parodė, kad tiek kompiuteris, tiek žmogus prastą vertimą pateikusias vertimo sistemas įvertino žemais balais, o gerą – aukštais. Buvo gauta aukšta koreliacija tarp skirtingais būdais gautų sistemų įvertinimo rezultatų. Būtent dėl šios aukštos koreliacijos tarp žmogaus ir kompiuterio įvertinimų *BLEU* tebelieka populiariausia automatine vertimo įvertinimo metrika (Koehn 2010: 217-224, 227, 230).

2. Mišriųjų automatinio vertimo sistemų analizė

Mišriosios automatinio vertimo sistemos – tai kelių tokių sistemų integracija. Tokios sistemos siūlo tikslesnį vertimą, nes:

- mišriosios sistemos yra kuriamos tokiu būdu, kad dviejų tipų sistemų kombinacija sudėtų pavienių sistemų privalumus, bet ne trūkumus (Carl *et al.* 2002). Pavyzdžiui, lingvistinė informacija, esanti taisyklių sistemoje, gali papildyti nepakankamas leksines statistinės sistemos žinias, kurios ypač pasireiškia, kai verčiami tekstai yra kitos dalykinės srities nei lygiagrečių tekstų bazės tekstai;
- autoriaus bakalauro baigiamajame darbe buvo parodyta, kad statistinių vertimo sistemų daugiaprasmiškumo problemą galima bandyti spręsti naudojant taisyklių rinkinį;
- įvairūs užsienio autoriai [Chen *et al.* 2010; Dugast *et al.* 2007; Eisele *et al.* 2008; Matusov *et al.* 2009] savo straipsniuose parodė, kad jų pasiūlytos mišriosios vertimo sistemos pateikė tikslesnį vertimą už pavienių sistemų išverstą tekstą;
- vienos iš seniausių taisyklėmis paremtų automatinio vertimo sistemų *SYSTRAN* kūrėjai jau keletą metų naudoja ir statistinius metodus (Marin 2010:16).

2.1. Automatinio vertimo sistemų integracijos būdai

Mišriosios automatinio vertimo sistemos yra kelių sistemų integracija. Egzistuoja du tokių sistemų integracijos būdai (Eisele 2007):

Gilioji integracija. Tokia integracija pasiekiama sukūrus naują architektūrinį modelį, kuriame sistemos, paremtos vienu vertimo principu, pranašumai įtraukiami į kitu principu paremtą sistemą. Pavyzdžiui, tai gali būti tokios sistemos:

- Sistema, paremta taisyklėmis ir papildyta mokymosi moduliu, kuris leidžia sistemai išmokti naujų taisyklių.
- Statistinė vertimo sistema, į kurią įtraukti sintaksės apribojimai arba taisyklės.

Paviršutiniška integracija. Tai tiesiog dviejų ar daugiau automatinio vertimo sistemų sujungimas į vieną sistemą. Tokios sistemų integracijos pavyzdžiai gali būti:

- Taisyklėmis paremtos sistemos, kurių rezultatai apdorojami statistika. Tokiu atveju vertimas atliekamas naudojant taisyklėmis paremtą variklį. Tuomet panaudojami statistiniai metodai, kurie bando pakoreguoti gautą rezultatą.

- Statistika paremtos sistemos, valdomos taisyklėmis. Šiuo atveju taisyklės gali būti naudojamos pradiniam teksto apdorojimui, siekiant geriau išnaudoti statistinį variklį. Taisyklės taip pat gali apdoroti galutinį statistinį rezultatą atlikdamos tokias funkcijas kaip normalizacija.

Gilioji vertimo sistemų integracija pateikia geresnį vertimą, tačiau yra sunkiau įgyvendinama nei paviršutiniška integracija. Šiuo metu žymiausi giliosios integracijos vertimo modeliai yra plėtojami Edinburgo ir Prahos universitetuose, o paviršutiniškos integracijos tyrimai atliekami Vokietijos Saarlando universitete.

2.1.1. Sumaišymo tinklai

Automatinio vertimo sistemose įvedamas tekstas šiuo metu yra paprastos žodžių sekos formos. Tačiau kyla paklausa integruoti tokias sistemas į sudėtingą informaciją apdorojančias sistemas, tokias kaip, pavyzdžiui, kalbos atpažinimo sistema. Čia generuojama daug hipotezių su įvairiu klaidingumo laipsniu. Šiuolaikinės automatinio vertimo sistemos apdoroja tik vieną įvesties hipotezę, todėl atsirado žodinė gardelė (angl. *word lattice*), kurią panaudojus galima apdoroti daug netikslių hipotezių ir pateikti geriausią vertimą. Kadangi tokia gardelė yra sudėtingos struktūros, praktikoje naudojamas jos supaprastintas variantas – sumaišymo tinklas (angl. *confusion network*). Tokio tinklo gavimas iš gardelės yra padaromas su tokiais įrankiais kaip *SRILM*.

Sumaišymo tinklas – tai kryptingas grafas su svoriais, kurio ypatybė yra ta, kad kiekvienas šio grafo kelias pereina per visus kitus grafo mazgus. Kiekvienas lankas yra pažymėtas žodžiu ir tikimybe. Lankų tikimybių tarp dviejų mazgų suma lygi vienetui. Bet koks kelias sumaišymo tinkle yra šio grafo realizacija. Šios realizacijos gali skirtis žodžių tvarka arba bendru rezultatu (tikimybe).

Pavyzdyje, pateiktame 12 pav., pavaizduoti trijų skirtingų vertimo sistemų vieno sakinio vertimai. Čia kiekvieno sumaišymo tinklo žodžio galutinio varianto tikimybė yra skaičiuojama paprasčiausiu įmanomu būdu – sumuojant kiekį kartų, kuriuos žodis aptinkamas trijuose pateiktuose vertimuose ir dalijant šitą kiekį iš vertimų skaičiaus (trijų). Vis dėlto siūloma naudoti tobulesnius žodinių tikimybių būdus, pavyzdžiui, tokius, kurie yra siūlomi straipsnių (Matusov *et al.* 2009; Rosti *et al.* 2008] autorių. Akivaizdu, kad kai kurios vertimo sistemos yra labiau patikimos negu kitos, todėl jos labiau lemia žodžio pasirinkimo tikimybę. Taip pat vertėtų įtraukti ir kalbos modelio priklausomybes tam, kad išverstas sakinys pasižymėtų sklandžia kalba.

Sistemų vertimai

Šitą sakinį išvertė mašininio vertimo sistema

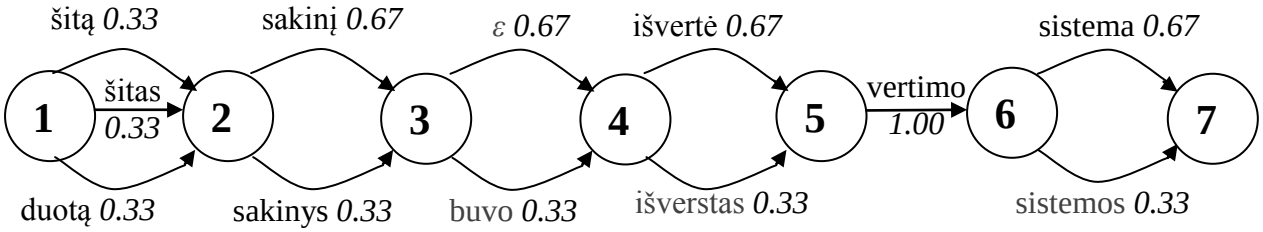
Šitas sakinys buvo išverstas automatinio vertimo sistemos

Automatinio vertimo sistema išvertė duotą sakinį

Sumaišymo tinklas

Šitą 0.33	sakinį 0.67	ε 0.67	išvertė 0.67	mašininio 0.33	vertimo 1.00	sistema 0.67
Šitas 0.33	sakinys 0.33	buvo 0.33	išverstas 0.33	automatinio 0.67		sistemos 0.33
Duotą 0.33						

12 pav. Sumaišymo tinklas paverčiantis trijų skirtingų sistemų vertimus alternatyvių žodžių variantų seka (įskaitant epsilon arba tuščią simbolį), kas leidžia pasiekti vieningą vertimą



13 pav. Grafinis sumaišymo tinklo vaizdavimas

Sumaišymo tinklo konstravimas iš kelių sistemų vertimų nėra trivialus uždavinys. Tai galima pastebėti 12 paveiksle, kur lietuviškų žodžių poros „šitą“ ir „duotą, „mašininio“ ir „automatinio“ atsidūrė tuose pačiuose stulpeliuose. Tokį sprendimą integruota vertimo sistema priėmė remdamasi žodžių priskyrimo metodais, kurie taip pat naudojami ir vertimo modeliui sudaryti. Šiuo konkrečiu atveju reikėtų skaičiuoti tikimybes $p(e_m \mid e_n)$, tai yra tikimybes, kad paėmus tam tikros sistemos vertimo hipotezę e_n kaip pagrindą arba karkasą (angl. *skeleton*), tikėtina hipotezė e_m.

Realus sumaišymo tinklo pritaikymo pavyzdys yra pateikiamas 4 lentelėje. *LKI* ir „*Tilde IT*“ vertimo sistemos tinklalapyje išvesties lange yra pateiktas sakinys „Mašininio vertimo rezultatai padeda suprasti teksto prasmę, bet nėra žmogaus vertėjo pakaitalas.“ Iš trijų sakinio vertime dalyvavusių vertimo sistemų „*Tilde IT*“ pateikė prasčiausią vertimą, kurį galima išversti kaip „Mašininio vertimo rezultatai padeda suprasti tekstą, bet ne žmones vertėjus“. Sistemos *Bing Translator* vertimas buvo pasirinktas kaip šio pavyzdžio karkasas. Ši sistema padarė tik vieną klaidą – pateikė trečiojo asmens veiksmažodžio formą (žodis „*helps*“) daugiskaitai (žodis „*results*). *Google Translate* sistema taip pat padarė vieną nežymią klaidą – panaudojo artikelį „*the*“ vietoje teisingo ”*a*“. Ketvirtajame ir penktajame 4 lentelės stulpeliuose pateikti sumaišymo tinkle dalyvaujantys žodžiai. Kiekvienam iš jų buvo priskirta tikimybė remiantis tuo pačiu principu, kuris buvo naudotas 12 pav. pavaizduotame pavyzdyje. Paskutiniame stulpelyje

pateikiamas galutinis vertimo variantas, kuris gaunamas paimant didžiausią tikimybę turinčius sumaišymo tinklo žodžius. Šiame pavyzdyje deja negalima įžvelgti dar vienos problemos konstruojant sumaišymo tinklą, kuri yra žodžių tvarka sakiniuose. Grįžtant prie pavyzdžio, pateikto 12 pav., galima pastebėti, kad trečiajame vertimo variante žodžiai „automatinė vertimo sistema“ yra sakinio pradžioje, o kituose sakiniuose ši frazė yra pabaigoje. Galima manyti, jei sumaišymo tinklas leidžia gauti vertimą, kuriame žodžių pasirinkimas skiriasi nuo bet kurios vienos sistemos vertimo žodžių pasirinkimo, tas pats turėtų galioti ir žodžių tvarkai. Tačiau praktikoje iš esmės naudojami paprastesni metodai:

- Naudoti, jūsų nuomone, patikimiausios vertimo sistemos vertimo žodžių tvarką kaip karkasą. Toliau reikėtų derinti visus kitus vertimus prie šito sakinio.
- Antrasis variantas analogiškas pirmajam, tik karkasą rinktis iš naujo sulig kiekvienu nauju verčiamu sakiniu. Tokiu būdu gautume vienos sistemos vertimą, kuris labiausiai panašus į visų kitų sistemų vertimus.

Taigi sumaišymo tinklas sudaro galimybę įgyvendinti vertimo sistemų paviršutinišką integraciją. Keleto sistemų pateiktas vertimas, pataisytas pagal šį tinklą, leidžia gauti tikslesnį vertimą, ką įrodė E. Matusovas ir kiti autoriai savo straipsnyje (Matusov *et al.* 2009).

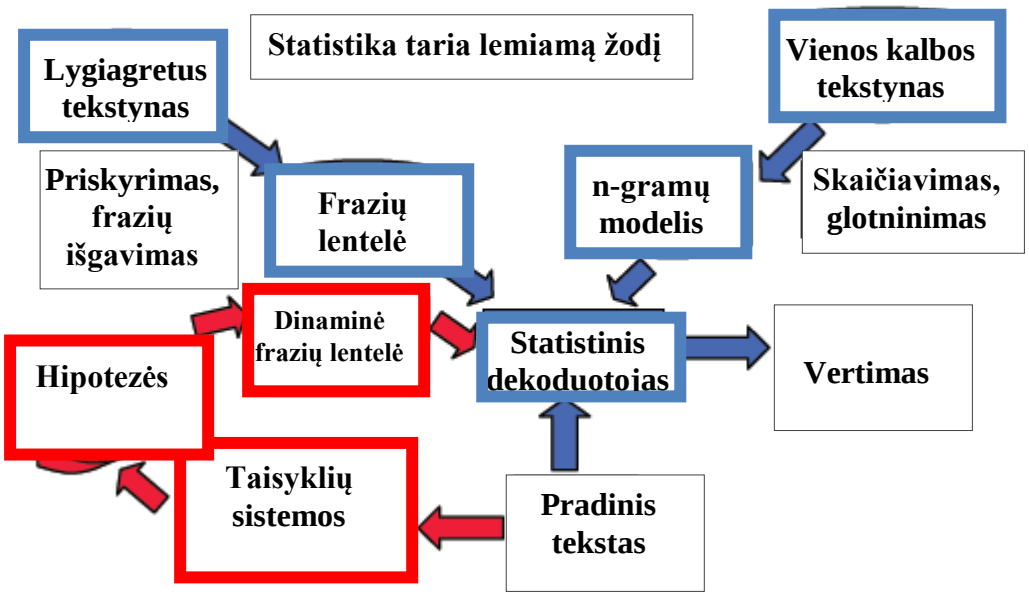
4 lentelė. Sumaišymo tinklo konstravimas remiantis trimis įvairių sistemų pateiktais vieno sakinio vertimo variantais

Pradinis tekstas	***Google Translate***	***Bing Translator***	***„Tilde IT“* ir *LKI***	**Sumaišymo tinklas**		**Galutinis sakinio variantas**
				Žodžių pirmieji variantai	**Žodžių antrieji variantai**	
Mašininio	Machine	Machine	Machine	Machine **1**		Machine
vertimo	translation	translation	translation	translation **1**		translation
rezultatai	results	results	results	results **1**		results
padeda	help	helps	help	helps **0.33**	help **0.67**	help
suprasti	to	to	you	to **0.67**	you **0.33**	to
teksto	understand	understand	understand	understand **1**		understand
prasmę	the	the	the	the **1**		the
bet	meaning	meaning	text	meaning **0.67**	ε **0.33**	meaning
nėra	of	of	but	of **0.67**	ε **0.33**	of
žmogaus	the	the	not	the **0.67**	ε **0.33**	the
vertėjo	text	text	the	text **1**		text
pakaitalas	but	but	human	but **1**		but
	there	there	translators	there **0.67**	ε **0.33**	there
	is	is		is **0.67**	ε **0.33**	is
	no	no		no **0.67**	not **0.33**	no
	substitute	substitute		substitute **0.67**	ε **0.33**	substitute
	for	for		for **0.67**	ε **0.33**	for
	the	a		a **0.33**	the **0.67**	the
	human	human		human **1**		human
	translator	translator		translator **0.67**	translators **0.33**	translator

2.2. Mišriųjų automatinio vertimo sistemų architektūros

Pastaraisiais metais buvo pasiūlytos kelios perspektyvios automatinio vertimo sistemų integracijos architektūros. Galima ypač išskirti Europos mokslininkų, dalyvaujančių *EuroMatrix* projekte, pasiekimus. Šiuo metu vykdoma jau antroji šio Europos Sąjungos remiamo projekto stadija *EuroMatrixPlus*. Šio projekto vieni pagrindinių dalyvių vokiečių Saarlando universiteto mokslininkai viename iš savo straipsnių (Chen *et al.* 2010) siūlo savo integracijos architektūrą, kuri leidžia integruoti statistinę ir vieną ar kelias taisyklėmis paremtas automatinio vertimo sistemas. Centrinis tokios architektūros elementas yra atvirojo kodo *Moses* statistinė vertimo sistema (sukurta *EuroMatrix* projekto dalyvių) su modifikuota frazių lentele. Standartinė statistinės sistemos frazių lentelė yra papildoma įrašais, gautais verčiant tekstus įvairiomis taisyklių sistemomis, ir ja galima rasti tikslesnes frazių kombinacijas. Taikant standartinius programinės įrangos modulius *GIZA++* žodinių blokų identifikavimui ir *Moses* jų rekombinacijai, kyla sunkumų dėl išverstų fragmentų žodžių tvarkos. Tačiau siūloma architektūra suteikia galimybę paprastai išspręsti šią sudėtingą kelių vertimo sistemų rezultatų integracijos problemą. Galutinį vertimo parinkimą iš praplėstos frazių lentelės atlieka statistinės vertimo sistemos *Moses* dekoduotojas, kuris ieško geriausio vertimo rekombinuodamas žodinius blokus, gautus iš taisyklėmis paremtų sistemų ir originalios statistinės vertimo sistemos (apmokytos pagal *Europarl* tekstynus).

Sistemos architektūra pavaizduota 14 pav., kur mėlynai apibrėžtos dalys atvaizduoja statistinę vertimo sistemą, o raudonai – modulius ir duomenų rinkinius, gautus iš taisyklėmis paremtų sistemų.



14 pav. Y. Chen ir A. Eisele pasiūlyta mišriosios automatinės vertimo sistemos architektūra

Pasiūlytai mišriajai automatinio vertimo sistemai įvertinti autoriai suskaičiavo *BLEU* reikšmes šešioms visoms kalbų poroms (vokiečių-anglų, anglų-vokiečių, prancūzų-anglų, anglų-prancūzų, ispanų-anglų, anglų-ispanų), taip pat ir visoms šešioms naudotoms taisyklių sistemoms bei statistinei sistemai, jei jos būtų taikomos kaip pavienės vertimo sistemos (5 lentelė). Remiantis gautais rezultatais, mišrioji sistema siūlo tikslesnį vertimą visų kalbų porų atveju, jei verčiamo teksto dalykinė sritis sutampa su turimų tekstynų sritimi. Kitais atvejais, pavyzdžiui, verčiant publicistinius naujienų tekstus geresnė *BLEU* reikšmė pasiekiama tik dviem kalbų poroms (ispanų-anglų ir anglų-ispanų).

5 lentelė. Y. Chen ir A. Eisele pasiūlytos mišriosios vertimo sistemos, šešių taisyklėmis paremtų sistemų ir statistinės vertimo sistemos *Moses* vertimų *BLEU* metrikos reikšmės

	Tekstynas *Europarl*						Tekstynas *NewsCommentary*					
	vokiečių –> anglų	anglų –> vokiečių	prancūzų –> anglų	anglų –> prancūzų	ispanų –> anglų	anglų –> ispanų	vokiečių –> anglų	anglų –> vokiečių	prancūzų –> anglų	anglų –> prancūzų	ispanų –> anglų	anglų –> ispanų
SMT	22.81	19.78	24.18	21.62	31.68	24.46	14.24	9.75	11.60	12.24	17.27	14.48
Hybrid	**27.85**	**20.75**	**28.12**	**28.82**	**33.15**	**32.31**	17.36	13.57	17.66	20.71	**22.16**	**22.55**
RBMT1*	13.34	11.09	——	17.19	——	18.63	14.90	12.34	——	15.11	——	17.13
RBMT2	16.19	12.06	——	——	——	——	16.66	13.64	——	——	——	——
RBMT3	16.32	10.88	18.18	20.38	19.32	20.89	16.88	12.53	17.20	18.82	19.00	19.98
RBMT4	15.58	12.09	19.00	22.20	18.99	21.69	**17.41**	**13.93**	17.73	**20.85**	19.14	21.70
RBMT5	15.58	9.54	21.36	12.98	18.47	20.59	15.99	11.05	**18.65**	19.49	20.50	20.02
RBMT6	13.96	9.44	17.16	18.91	18.01	19.18	15.08	10.41	16.86	17.82	18.70	19.97

Kanados kompiuterinės lingvistikos mokslininkai (Simard *et al.* 2007) pasiūlė tokią paviršutiniškos integracijos sistemą, kur panaudojamas išvedamo teksto automatinis postredagavimas (angl. *automatic post-editing* arba *APE*). Pradinis tekstas yra išverčiamas į kitą kalbą naudojant taisyklėmis paremtą vertimo sistemą *SYSTRAN*, tuomet gautas tekstas yra redaguojamas taikant statistinę sistemą *PORTAGE* (Kanados tyrimų tarybos produktas). Tokią sistemų integraciją atlikti autorius paskatino tų pačių tipų klaidos, kurias daro taisyklėmis paremtos sistemos, o tinkamai apmokius statistinę sistemą, tokių klaidų galima išvengti. Naudojama *PORTAGE* statistinė vertimo sistema gaunamą *SYSTRAN* vertimą supranta kaip įvesties kalbą, o išversto teksto teisingą versiją (išverstą profesionalaus vertėjo) – kaip kalbą, ketinamą gauti po įvesties redagavimo. Toks vertimo procesas gali būti apibūdintas kaip automatinis taisyklėmis paremtos vertimo sistemos pritaikymas konkrečiai dalykinei sričiai, kadangi dažniausiai statistinė sistema bus apmokoma remiantis konkrečios srities tekstais.

Autorių apskaičiuotos *BLEU* reikšmės prancūzų-anglų ir anglų-prancūzų kalbų poroms yra pavaizduotos 6 lentelėje.

6 lentelė. M. Simardo ir kitų pasiūlytos mišriosios sistemos *BLEU* reikšmės prancūzų-anglų ir anglų-prancūzų kalbų poroms

	anglų –> prancūzų	prancūzų –> anglų
Tekstynas *Europarl* (>32 milijonų žodžių vienai kalbai)		
SYSTRAN	23.06	20.11
PORTAGE	31.01	30.90
SYSTRAN+PORTAGE	31.11	30.61
Tekstynas *NewsCommentary* (1 milijonas žodžių vienai kalbai)		
SYSTRAN	24.41	18.09
PORTAGE	25.98	25.17
SYSTRAN+PORTAGE	28.80	26.79

Pateikti rezultatai leidžia teigti, kad *APE* žymiai padidina taisyklėmis paremtos sistemos *SYSTRAN BLEU* reikšmes. Šis prieaugis yra ypač pastebimas naudojant *Europarl* srities tekstus ir verčiant į anglų kalbą. Publicistinių naujienų tekstų atveju *APE* strategija (*SYSTRAN+PORTAGE* eilutės) akivaizdžiai pranoksta tiesioginį statistinį vertimą (*PORTAGE* eilutės). Verčiant į anglų kalbą, *BLEU* reikšmė yra 1.5 balo, o į prancūzų – 3 balais didesnė, o *Europarl* srities tekstų vertimas demonstruoja beveik vienodas *BLEU* reikšmes. Taip yra dėl to, kad *Europarl* tekstyne buvo panaudota mažiau nei 50 tūkstančių sakinių porų, kas yra 30 kartų mažiau nei tikrasis *Europarl* tekstyne esantis sakinių skaičius. Taigi autoriai prieina išvadą, kad *APE* geriausia naudoti sritims su apribotų kiekių prieinamų apmokymo duomenų. Autoriai pabrėžia, kad *APE* strategija žymiai pagerina *SYSTRAN* sistemos vertimą, panaudojus tik 5 tūkstančius sakinių porų vertimui iš anglų į prancūzų ir tik 2 tūkstančius – iš prancūzų į anglų.

Tolesni jų tyrimai pademonstravo, kad didinant *Europarl* apmokymo duomenų kiekį, statistinė vertimo sistema *PORTAGE* ir *SYSTRAN* sistema su *APE* strategija gerina vertimo kokybę logaritmiškai. Tačiau statistinės vertimo sistemos vertimo kokybės reikšmės susilygina su *SYSTRAN* sistema tam tikrame intervalo tarp 100 tūkstančių ir vieno milijono sakinių porų esančiame taške.

Dabar galima palyginti M. Simardo ir kitų pasiūlytos sistemų integracijos varianto *BLEU* reikšmes (6 lentelė) anglų-prancūzų ir prancūzų-anglų poroms su Y. Cheno ir A. Eisele pasiūlyta mišriąja sistema (5 lentelė). M. Simardo ir kitų sistema pranoksta kitą sistemą tiek verčiant *Europarl* srities tekstus, tiek publicistinius naujienų tekstus (čia pranašumas ypač didelis – virš aštuonių *BLEU* balų).

Vokietijos mokslininkai savo straipsnyje (Matusov *et al.* 2009), kuris pateko į žymaus *MIT* instituto 2009 m. išleistą knygą (geriausių straipsnių rinkinį), pasiūlė vertimo sistemų integracijos architektūrą, pagal kurią pasiekiamas konsensusas tarp visų dalyvaujančių sistemų išverstų tekstų. Konsensusas pasiekiamas pasitelkiant jau minėtą specialų kryptinį aciklinį grafą, vadinamą sumaišymo tinklu ir jam pritaikant daugumos balsavimą su svoriais. Skirtingos žodžių tvarkos tarp vertimų problemai išspręsti autoriai siūlo statistiniu modeliavimu paremtą žodžių priskyrimo procedūrą, kuri išmoksta žodžių atitiktis tarp kelių vertimo variantų. Siūloma sistemų integracija buvo išbandyta su kinų-anglų ir arabų-anglų kalbų poromis. Iš viso buvo naudojamos penkios pažangios automatinio vertimo sistemos.

E. Matusovo ir kitų siūloma vertimo sistemų architektūra pateikiama 15 pav. Toliau išvardijami konkretūs visų dalyvaujančių vertimo sistemų išvedamo teksto žodžių priskyrimo strategijos algoritmo etapai:

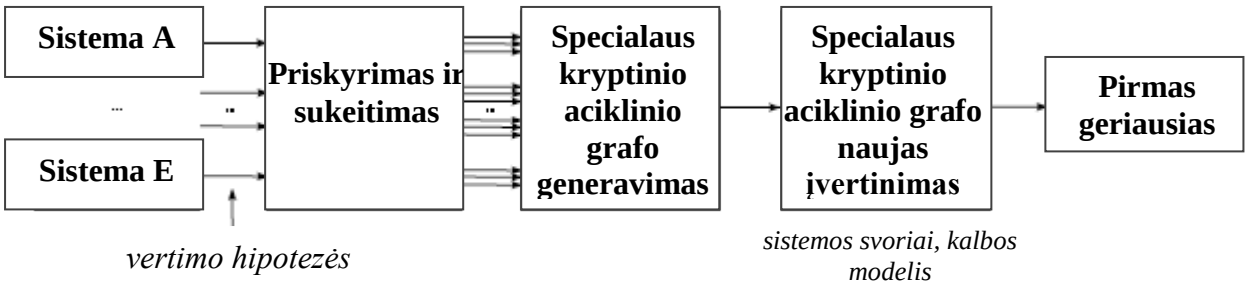
1. Tarp individualių mašininių vertimų hipotezių išrinkti pirminę (karkasą), kuri kaip manoma turi teisingiausią žodžių tvarką.

2. Atlikti pirminės ir likusių (antrinių) hipotezių žodžių priskyrimo operaciją. Ši operacija dažniausiai yra nemonotoninė (galinti apdoroti skirtingos sintaksės hipotezes).

3. Sukeisti žodžius antrinėse hipotezėse tokiu būdu, kad jų išsidėstymas būtų kuo panašesnis į pirminės hipotezės žodžių išsidėstymą.

4. Sukurti vertimo alternatyvų kiekvienam žodžiui pirminėje hipotezėje sumaišymo tinklą. Į šį tinklą įtraukti įterptinus ir trintinus žodžius.

5. Įvertinti kiekvieną sumaišymo tinklo lanką įvairių tipų statistiniais modeliais ir išrinkti geriausią žodžių seką naudojant daugumos balsavimą su svoriais (tai yra geriausią kelią grafe).



15 pav. E. Matusovo, G. Leuscho ir H. Ney pasiūlytos vertimo sistemų integracijos architektūra

Gauti sistemų integracijos rezultatai pateikė žymiai pagerinto vertimo kiekvienai tirtai kalbų porai. Pavyzdžiui, sistemos *BLEU* reikšmė verčiant iš kinų į anglų kalbą pagerėjo 3.1%, o iš arabų į anglų - 1.8%.

Lietuvoje kol kas vienintelis darbe nagrinėjama tema parašytas straipsnis, kurį pavyko rasti, yra A. Laukaičio ir O. Vasileco darbas (Laukaitis *et al.* 2007). Savo darbe autoriai pateikia asimetrinę (dėl lingvistinių kalbų resursų nevienodumo) mišriosios vertimo sistemos architektūrą ir ją pritaiko anglų-lietuvių kalbų porai. Anglų kalbą yra apdorojama (abstrahuojama) iki sakinio gramatinių dalių su *GATE* programine įranga. Darbe taip pat panaudoti žiniatinklio paieškos agentai (angl. *crawlers*), kuriais buvo surinktas tekstynas iš abiem kalbom parašytų tekstų fragmentų. Autoriai teigia, kad pasiekus pusiausvyrą tarp tinkamo prieinamų tekstinių duomenų abiem kalbom kiekio ir anglų kalbos abstrahavimo lygio galima tikėtis daug geresnės vertimo *BLEU* vertės padidėjimo.

3. Mišriosios automatinio vertimo sistemos sukūrimas

Prieš sukuriant mišriąją automatinio vertimo sistemą reikia nuspręsti kokios sistemos bus naudojamos ir kuria kryptimi bus verčiama. Buvo nuspręsta pasirinkti vertimo iš lietuvių į anglų kalbą kryptį. Toki pasirinkimą galima pagrįsti tokiais teiginiais:

- anglų kalba yra daugelio veiklos sričių oficiali kalba, todėl natūralu, kad šia kalba prieinamų duomenų kiekis yra nepalyginamai didesnis už kompiuterizuotu pavidalu prieinamų kitų kalbų duomenų kiekį. Tai yra svarbu, nes verčiant į anglų kalbą galima naudotis įvairiais internetiniais lingvistiniais įrankiais ir žodynais, kurių lietuvių kalbai kol kas trūksta. Be to, kalbos modelis gali būti papildytas bet kokiu norimu angliškos informacijos kiekiu sklandžiam vertimui gauti;
- vertimas iš anglų į lietuvių kalba dažniausiai naudojamas angliškos informacijos esmei suvokti (angl. *gisting*). Šiam tikslui pasiekti jau egzistuoja keli įrankiai, kurie buvo apžvelgti darbo 1.4 poskyryje. Naudotojas, kaip savo gimtosios kalbos žinovas, reikalui esant turi galimybę pataisyti nesklandų sistemos vertimą. Tačiau verčiant į anglų kalbą (kas yra labai aktualu organizacijoms, žmonėms rašantiems straipsnius ir pan.) susiduriama su sunkumais, susijusiais su kalbos ypatumų nežinojimo. Pavyzdžiui, dažnai yra bandoma versti tekstą pažodžiui, nesusimąstant apie tai, kad anglakalbiai žmonės gali apibūdinti konkrečius dalykus ar situacijas tik jų kalbai būdingomis konstrukcijomis (reiškinys vadinamas lingvistiniu reliatyvumu).

Mišriąją vertimo sistemą buvo nuspręsta kurti integruojant keturias vertimo sistemas: *Google Translate, Bing Translator, LKI ir Tilde IT* ir *Moses*. *Moses* vertimo sistemos kūrimas aprašomas kitame poskyryje. Taip pat visos sistemos turėjo būti įvertintos, nes norint naudoti sumaišymo tinklą integracijai, reikia pasirinkti karkasą – šiuo atveju sistemą, kuri pateikė geriausius rezultatus verčiant iš lietuvių į anglų kalbą. Sistemų įvertinimo rezultatai pateikiami 3.2 poskyryje. Galiausiai yra aprašoma vertimo sistemų integracijos metodika ir jos praktinis pritaikymas.

3.1. Automatinio vertimo sistemos *Moses* sukūrimas

Pirminių bandymų tikslas buvo sukurti veikiančią statistinio vertimo sistemą *Moses*, kuri ir sudarys mišriosios vertimo sistemos pagrindą. Pradiniai bandymai, kurių metu panaudotas *Europarl 6* lygiagretus lietuvių ir anglų kalbų tekstynas, padėjo sukurti kalbos ir vertimo modelius pagal 450 tūkstančių sakinių abiem kalbomis. Tačiau dėl nepakankamo operatyviosios atminties kiekio testavimo kompiuteryje nebuvo įmanoma patikrinti sukurtų modelių. Tai pavyko padaryti pakartotinų bandymų metu, kai buvo panaudotas spartesnis kompiuteris su didesniu kiekiu operatyviosios atminties. Visi

bandymai buvo atliekami Linux operacinėje sistemoje *OpenSUSE 11.4* (vėliau *OpenSUSE 12.1*). Šiame skyriuje plačiau aptariami etapai, kurie buvo įvykdyti, kad sistema būtų sėkmingai sukurta.

Pirmasis *Moses* sistemos sūkurio etapas buvo reikiamų **įrankių ir duomenų atsisiuntimas**. *Moses* turi pakankamai nemažą kiekį scenarijų, kuriems yra būtini tam tikri išoriniai įrankiai:

- visų pirma, tai *GIZA++* statistinio vertimo įrankių rinkinys;
- *mkcls* – įrankis žodžių klasėms gauti;
- *tcl/tk* – bendros paskirties programavimo kalba, kuri būtina norint įdiegti *Moses*;
- *SRILM* įrankiai – statistinės priemonės kalbos modeliams kurti ir yra būtini;
- *IRSTLM* įrankiai yra nebūtini, tačiau kadangi jie siūlo tokią savybę kaip kalbos modelių veikimas diskinėje atmintyje (vietoje operatyviosios, kurios gali neužtekti didelių modelių įkėlimui), rekomenduotina juos įdiegti.

Taigi prieš pradedant instaliuoti *Moses* statistinę vertimo sistemą, reikėjo atsisiųsti ir sukompiliuoti visus ankščiau išvardytus įrankius. Katalogas, kuriame buvo instaliuojami visi įrankiai, yra `/home/igor`:

```
mkdir moses
cd moses
```

GIZA++ ir *mkcls* parsisiuntimas ir išpakavimas:

```
wget http://giza-pp.googlecode.com/files/giza-pp-v1.0.5.tar.gz
tar -xzvf giza-pp-v1.0.2.tar.gz
cd giza-pp
make
cd ../
mkdir bin
cp giza-pp/GIZA++-v2/GIZA++ bin/
cp giza-pp/mkcls-v2/mkcls bin/
cp giza-pp/GIZA++-v2/snt2cooc.out bin/
```

SRILM įrankiams gauti buvo reikalinga registracija, po kurios buvo gautas failas `srilm.tgz`:

```
mkdir srilm
cd srilm
tar -xzvf srilm.tgz
chmod +w Makefile
```

Failą *Makefile* reikia pataisyti taip, kad šios failo eilutės nurodytų absoliučius kelius:

```
> SRILM = /home/igor/moses/srilm
```

```
make World
```

IRSTLM įrankių instaliavimas:

```
mkdir irstlm
svn co https://irstlm.svn.sourceforge.net/svnroot/irstlm irstlm
cd irstlm
./install
```

Toliau buvo instaliuojamas pats *Moses*, kuris yra prieinamas *GitHub* tinklalapyje. Ši komanda nukopijuoja visą *Moses* programinį kodą į kompiuterį:

```
git clone git://github.com/moses-smt/mosesdecoder.git moses
```

Moses kompiliacija:

```
./regenerate-makefiles.sh
./configure --with-srilm = /home/igor/moses/srilm --with-irstlm =
/home/igor/moses/irstlm --with-giza = /home/igor/moses/bin -j2
```

Aukščiau pateiktoje eilutėje -j2 yra dviejų lygiagrečių užduočių veikimas (kompiuteriams su daugeliu procesorių branduolių, galima padidinti šį skaičių).

Taip pat buvo nustatytas sisteminis kintamasis, kurį naudojo tolesni scenarijai:

```
export SCRIPTS_ROOTDIR=/home/igor/moses/scripts
```

Taip pat buvo naudinga turėti kai kuriuos scenarijus, kurie nebuvo įtraukti į Moses:
wget http://homepages.inf.ed.ac.uk/jschroe1/how-to/scripts.tgz

```
tar -xzvf scripts.tgz
```

Galiausiai Moses vertimui įvertinti buvo reikalingas *NIST* įvertinimo įrankis:

```
wget ftp://jaguar.ncsl.nist.gov/mt/resources/mteval-v11b.pl
chmod +x mteval-v11b.pl
cp mteval-v11b.pl scripts
```

Antrasis etapas buvo **kalbos modelio sudarymas.** Kalbos modeliui sudaryti buvo panaudotas *Europarl 6* versijos lygiagretus lietuvių-anglų tekstynas. Šiame etape iš pradžių reikėjo atlikti pradinį šio tekstyno apdorojimą. Šiam tikslui buvo sukurtas darbinis katalogas su komanda *mkdir work*. Pirmasis tekstyno apdorojimo žingsnis yra sakinių išskaidymas į žodžius ir skyrybos ženklus (angl. *tokenization*):

```
mkdir work/corpus
gzip -cd data/training/lt-en.tgz | scripts/tokenizer.perl -l lt >
work/corpus/europarl.lt
gzip -cd data/training/lt-en.tgz | scripts/tokenizer.perl -l en >
work/corpus/europarl.en
```

Buvo svarbu filtruoti ilgus sakinius (simbolių ilgis > 40), kadangi dėl to apmokymas su GIZA++ labai užtruktų:

```
$SCRIPTS_ROOTDIR/clean-corpus-n.perl work/corpus/europarl.tok lt en
work/corpus/europarl.clean 1 40
```

Apmokymo duomenų raidžių pavertimas mažosiomis (angl. *lowercasing*) buvo atliktas šiomis komandomis:

```
scripts/lowercase.perl < work/corpus/europarl.clean.lt >
work/corpus/europarl.lowercased.fr
scripts/lowercase.perl < work/corpus/news-commentary.clean.en >
work/corpus/europarl.lowercased.en
```

Toliau buvo galima sukurti kalbos modelį:

```
mkdir work/lm
scripts/lowercase.perl < work/corpus/europarl.tok.en > work/lm/europarl.en
```

Trečios eilės n-gramų anglų kalbos modeliui sudaryti buvo naudojamas *SRILM* įrankis:

```
srilm/bin/i686/ngram-count -order 3 -interpolate -kndiscount -unk -text
work/lm/europarl.lowercased.en -lm work/lm/europarl.lm
```

Po kalbos modelio sudarymo reikėjo sudaryti patį vertimo modelį arba frazių lentelę su *GIZA++* įrankiu. Šį darbą atliko *Moses* scenarijus *train-model.perl*:

```
nohup nice $SCRIPTS_ROOTDIR/training/train-model.perl -scripts-root-dir
$SCRIPTS_ROOTDIR -root-dir work -corpus work/corpus/europarl.lowercased -f lt -e
en -alignment grow-diag-final-and -reordering msd-bidirectional-fe -lm
0:3:/home/igor/moses/work/lm/europarl.lm >& work/training.out &
```

Scenarijui atlikus darbą *work* kataloge buvo gautas failas *moses.ini*. Taip pat buvo gauti šie svarbūs failai:

- *extract.gz* – kataloge */home/igor/moses/work/model* esantis archyvas, kuriame yra frazių žodžių priskyrimo failas (kiekvienam lietuviškam žodžiui buvo priskirtas angliškas žodis arba *NULL* reikšmė);
- *extract.inv.gz* – kataloge */home/igor/moses/work/model* esantis archyvas, kuriame yra atvirkščias priskyrimo failas (kiekvienam angliškam žodžiui priskiriamas lietuviškas žodis arba *NULL* reikšmė);
- *en-lt.A3.final.gz* – kataloge */home/igor/giza.en-lt* esantis kiekvieno sakinio iš *Europarl* žodžių priskyrimas (figūriniuose skliaustuose pateikiami skaičiai, kurie reiškia kokie angliški žodžiai atitinka kiekvieną lietuvišką žodį sakinyje);
- *lt-en.A3.final.gz* – kataloge */home/igor/giza.lt-en* esantis kiekvieno sakinio iš *Europarl* žodžių priskyrimas (čia figūriniuose skliaustuose pateikiami skaičiai, kurie reiškia kokie lietuviški žodžiai atitinka kiekvieną anglišką žodį sakinyje).

Siekiant minimizuoti kompiuterio operatyviosios atminties naudojimą, kalbos modelis ir frazių lentelė buvo įdėtas į virtualiąją (diskinę) atmintį. Šią funkciją atliko *IRSTLM* įrankių rinkinys:

```
irstlm/bin/i686/compile-lm work/lm/europarl.lm work/lm/europarl.blm
cd work/lm
ln -s europarl.blm europarl.blm.mm
cd ../../
```

Frazių ir žodžių sukeitimo lenteles reikėjo surūšiuoti:

```
gzip -cd work/model/phrase-table.gz | LC_ALL=C sort | moses/misc/processPhraseTable -ttable
0 0 - -nscores 5 -out work/model/phrase-table
gzip -cd work/model/reordering-table.gz | LC_ALL=C sort | moses/misc/processLexicalTable -
out work/model/reordering-table
```

Toliau buvo padaryta work/model/moses.ini failo kopija ir šioje kopijoje naudojami failai buvo pakeisti į tuos, kurie naudoja virtualiąją atmintį:

```
cp work/model/moses.ini work/model/moses-bin.ini
```

Faile *moses-bin.ini* eilutės su kalbos ir vertimo modeliais turi būti pataisytos kaip parodyta žemiau:

```
1 0 0 5 /home/igor/moses/work/model/phrase-table
1 0 3 /home/igor/moses/work/lm/europarl-lten.blm.mm
```

Sistemos derinimo metu **testuojamų tekstų paruošimas** yra trečioji *Moses* sistemos sukūrimo dalis. Buvo panaudoti verslo ir teisės sričių tekstynai *Acquis Communitaire* (The JRC-Acquis Multilingual Parallel Corpus).

```
mkdir work/tuning
scripts/tokenizer.perl -l lt < data/dev/ac-dev.lt > work/tuning/ac-dev.tok.lt
scripts/tokenizer.perl -l en < data/dev/ac-dev.en > work/tuning/ac-dev.tok.en
mkdir work/evaluation
scripts/tokenizer.perl -l lt < data/devtest/ac-test.lt > work/evaluation/ac-
test.tok.lt
scripts/lowercase.perl < work/tuning/ac-dev.tok.lt > work/tuning/ac-
dev.lowercased.lt
scripts/lowercase.perl < work/tuning/ac-dev.tok.en > work/tuning/ac-
dev.lowercased.en
scripts/lowercase.perl < work/evaluation/ac-test.tok.lt > work/evaluation/ac-
test.lowercased.lt
```

Ketvirtoji *Moses* kūrimo stadija yra vadinama **sistemos derinimu** (angl. *tuning*). *Moses* vertimo sistema, turi 14 svorių, kurie vienaip ar kitaip lemia vertimą. Derinimo metu buvo rastos optimaliausios šių svorių reikšmės atlikus minimalaus klaidų kiekio apmokymus (angl. *minimum error rate training* arba

MERT). Šis procesas buvo atliekamas su į kietąjį diską įrašytomis modelių versijomis, tai yra panaudojus virtualiąją atmintį, o ne operatyviąją. Galiausiai po dviejų dienų trukusio nuolatinio apmokymo (buvo atlikta 18 *MERT* iteracijų) buvo surasti optimaliausi svoriai, kuriuos panaudojusi *Moses* vertimo sistema pasiekė 22,14 % *BLEU* įvertinimą. Sistemos derinimą atlieka šis scenarijus:

```
nohup nice $SCRIPTS_ROOTDIR/training/mert-moses.pl work/tuning/ac-
dev.lowercased.lt work/tuning/ac-dev.lowercased.en moses-cmd/src/moses
work/model/moses-bin.ini --working-dir work/tuning/mert --mertdir
/home/igor/moses/mert --rootdir $SCRIPTS_ROOTDIR --decoder-flags "-v 0" >&
work/tuning/mert.out &
```

Galiausiai buvo sukurtas failas work/tuning/mert/moses-bin.ini, kuriame yra optimaliausi svoriai. Šį failą reikėjo nukopijuoti į work/model katalogą:

```
cd ../../../
scripts/reuse-weights.perl work/tuning/mert/moses.ini < work/model/moses-bin.ini >
work/tuning/moses-tuned-bin.ini
```

3.2. Egzistuojančių lietuvių-anglų automatinio vertimo sistemų įvertinimas

Visos automatinio vertimo sistemos buvo įvertintos paskaičiavus kiekvienai iš jų *BLEU* metrikos reikšmes. *Moses* vertimo sistema pateikė tekstyno *Acquis Communitaire* failo *ac-test.lt* vertimą pagal šią komandą:

```
nohup nice moses-cmd/src/moses -config work/tuning/moses-tuned-bin.ini -input-file
work/evaluation/ac-test.lowercased.lt > work/evaluation/ac-test.tuned-bin.output
```

Kadangi *Moses* sistema pateikia visus vertimo žodžius mažosiomis raidėmis, buvo reikalinga atlikti kelis papildomus veiksmus. Pirmiausia scenarijus *train-recaser.perl* pagal *Europarl* tekstyną išmoksta žodžius, kurie prasideda didžiosiomis raidėmis. Antrasis scenarijus pritaiko šias žinias ir ištaiso paduotą vertimą:

```
scripts/recaser/train-recaser.perl -train-script scripts/training/train-model.perl
-ngram-count srilm/bin/i686/ngram-count -corpus work/corpus/europarl.tok.en -dir
/home/igor/moses/work/recaser -scripts-root-dir /home/igor/moses/scripts
scripts/recaser/recase.perl -model work/recaser/moses.ini -in work/evaluation/ac-
test.tuned-bin.output -moses moses/moses-cmd/src/moses > work/evaluation/ac-
test.tuned-bin.output.recased
```

Moses pateiktas vertimas taip pat yra pateikiamas išskaidytų simbolių pavidalu (pavyzdžiui, tarpai prieš ir po skyrybos ženklų), todėl buvo naudotas scenarijus *detokenizer.perl*, kuris pataiso vertimą:

```
scripts/detokenizer.perl -l en < work/evaluation/ac-test.tuned-bin.output.recased
> work/evaluation/ac-test.tuned-bin.output.detokenized
```

Norint įvertinti kiekvienos sistemos vertimą reikia sudaryti *XML* pavidalo failą. Žemiau pateikta komanda *Moses* vertimo sistemai (kitų sistemų vertimas paverčiamas *XML* pavidalu analogiškai, keičiami tik failų pavadinimai):

```
scripts/wrap-xml.perl data/devtest/ac-test-ref.en.sgm en lten <
work/evaluation/ac-test.tuned-bin.output.detokenized > work/evaluation/ac-
test.tuned-bin.output.sgm
```

Galiausiai automatinio vertimo sistemos pateikto vertimo įvertinimas atliekamas šia komanda:

```
scripts/mteval-v11b.pl -s moses/data/devtest/ac-test-src.lt.sgm -r
moses/data/devtest/ac-test-ref.en.sgm -t moses/work/evaluation/ac-test.tuned-
bin.output.sgm -c
```

Moses vertimo sistema pagal *BLEU-4* metriką gavo 22,14 % įvertinimą. Šiuo atveju buvo vertinami visų 4107 sakinių, esančių *Acquis Communitaire* tekstyne, vertimai. Visoms sistemoms palyginti buvo nuspręsta panaudoti tūkstantį tekstyno sakinių (žr. 7 lentelę).

7 lentelė. Vertimo sistemų įvertinimas pagal *BLEU* metriką remiantis tūkstančiu *Acquis Communitaire* sistemų išverstų sakinių

	BLEU-1	BLEU-2	BLEU-3	BLEU-4	BLEU-5	BLEU-6	BLEU-7	BLEU-8	BLEU-9
Google Translate	0.5968	0.4774	0.3964	**0.3354**	0.2866	0.2465	0.2134	0.1856	0.1622
Bing Translator	0.7316	0.6261	0.5491	**0.4893**	0.4398	0.3970	0.3594	0.3259	0.2962
LKI ir Tilde IT	0.7768	0.6969	0.6406	**0.5962**	0.5585	0.5249	0.4945	0.4667	0.4412
Moses	0.5525	0.3708	0.2603	**0.1893**	0.1417	0.1073	0.0815	0.0618	0.0458

Gauti įvertinimo rezultatai parodo, kad *LKI* ir *Tilde IT* vertimo sistema pagal standartinį *BLEU-4* įvertinimo būdą surinko net 59,62 %. Neblogą rezultatą parodė *Bing Translator* - 48,93 %, kitoms sistemoms sekėsi prasčiau: *Google Translate* – 33,54 %, *Moses* – 18,93 %. Tačiau drįstama teigti, kad gauti įvertinimai nėra teisingi. Labai tikėtina, kad dvi geriausiai pasirodžiusios vertimo sistemos buvo apmokytos remiantis testuojamais tekstynais. Tokiu atveju sistema „atsimena“ kiekvieną šio tekstyno sakinį ir pateikia tikslų tokių sakinių vertimą. Tai dar labiau patvirtina faktas, kad lietuvių ir anglų kalboms kitų lengvai prieinamų testavimo tekstynų, išskyrus *Acquis Communitaire*, tiesiog neegzistuoja.

Norint įsitikinti, kad gauti įvertinimo rezultatai nebuvo teisingi, buvo sukurtas dvidešimties sakinių tekstynas remiantis anglų ir lietuvių kalba parašytomis disertacijos (Rosinaitė 2010) dalimis. Buvo atliktas pakartotinas vertimo sistemų įvertinimas (žr. 8 lentelę). Šį kartą geriausią rezultatą parodė *Bing Translator* vertimo sistema, surinkusi 16,75 %, *Google Translate* sistema (13,65 %) pranoko praeito įvertinimo geriausią sistemą, sukurtą *LKI ir Tilde IT* ir *Moses*, kurios gavo panašius įvertinimus (10,18 % ir 9,92 % atitinkamai).

8 lentelė. Vertimo sistemų įvertinimas pagal *BLEU* metriką remiantis dvidešimt sistemų išverstų sakinių

	BLEU-1	BLEU-2	BLEU-3	BLEU-4	BLEU-5	BLEU-6	BLEU-7	BLEU-8	BLEU-9
Google Translate	0.5257	0.3253	0.2102	**0.1365**	0.0838	0.0563	0.0373	0.0000	0.0000
Bing Translator	0.5407	0.3551	0.2414	**0.1675**	0.1159	0.0812	0.0597	0.0437	0.0304
LKI ir Tilde IT	0.4664	0.2826	0.1641	**0.1018**	0.0626	0.0425	0.0311	0.0235	0.0176
Moses	0.4800	0.2699	0.1582	**0.0992**	0.0590	0.0000	0.0000	0.0000	0.0000

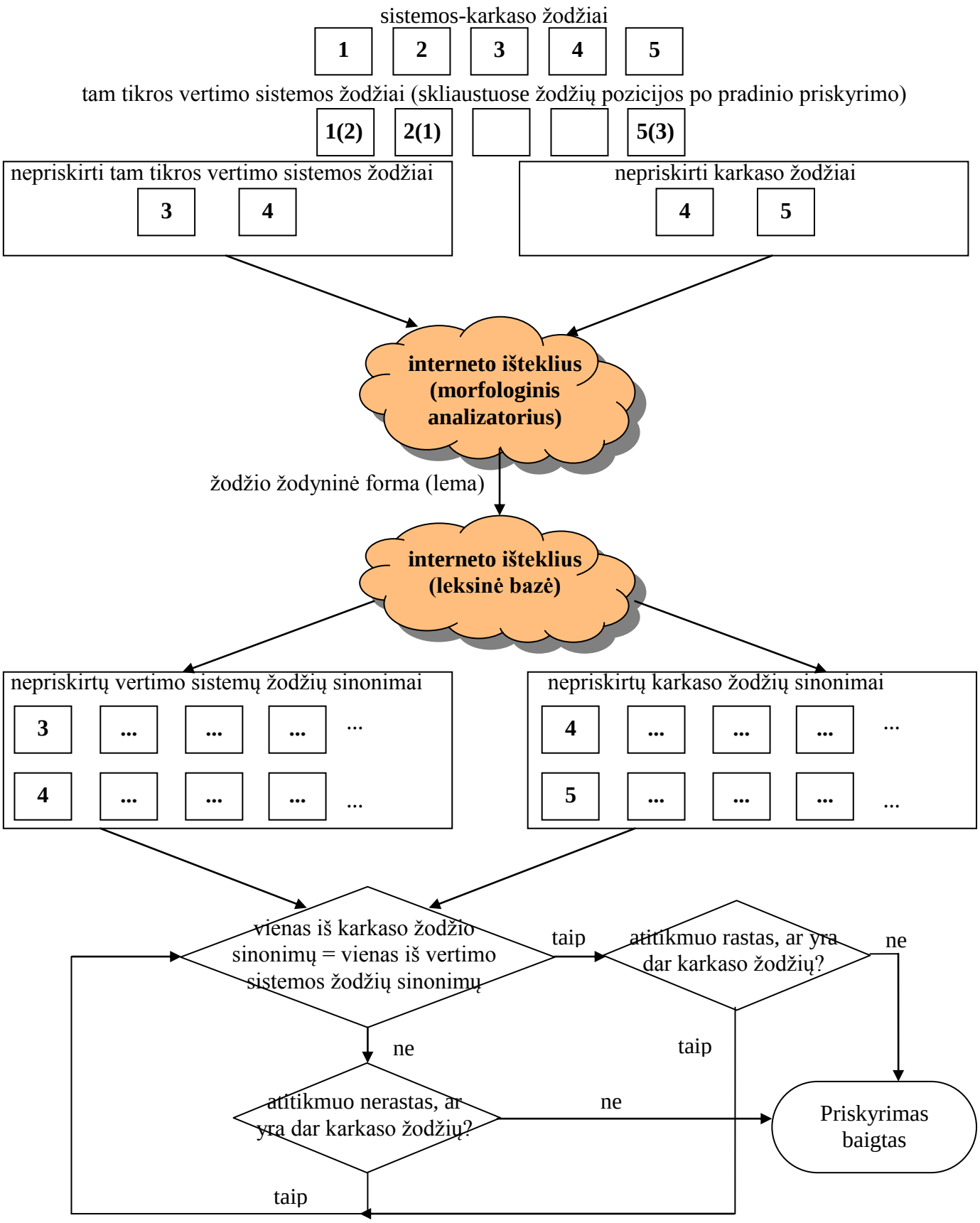
3.3. Automatinio vertimo sistemų integracija

Vertimo sistemų integracija buvo atliekama naudojant sumaišymo tinklą. Sukurta programa – tai keturių statistinių vertimo sistemų integracija. Panaudotos *Google Translate*, *Bing Translator*, *LKI ir Tilde IT* ir autoriaus sukurta *Moses* vertimo sistemos. Pirmosios trys sistemos buvo pajungtos panaudojant anksčiau pateiktas *HTTP* užklausas. Programai sukurti naudota *Java* programavimo kalba ir *Eclipse EE IDE* programavimo aplinka. Programos tipas yra servletas – programa, kurioje galima naudoti *HTML* kodą ir kurią galima paleisti kaip paslaugą. Vertimo sistema yra prieinama adresu: http://84.32.228.222:8080/vertimas/si. Programos grafinė naudotojo sąsaja yra pateikta 17 pav.

Kaip buvo paminėta sumaišymo tinklo konstravimas nėra trivialus uždavinys. Pagrindinės dvi problemos, kurias reikėjo išspręsti yra skirtingų vertimo variantų žodžių priskyrimas ir žodžių tvarkos galutiniame vertimo variante nustatymas.

Sudėtingas žodžių tvarkos nustatymo galutiniame vertimo variante uždavinys yra išspręstas tik dalinai. Galutinei žodžių tvarkai nustatyti taikomas minimalaus atstumo metodas. Pavyzdžiui, jeigu vienos sistemos vertime pateiktas unikalus žodis yra penktoje pozicijoje tai galutiniame variante jis bus tokioje pozicijoje, kuri yra arčiausia penktajai.

Žodžių priskyrimo problemai spręsti siūlomas naujoviškas sprendimas **– naudoti interneto išteklius**, tokius kaip *Princetono* universiteto leksinė anglų kalbos bazė (Wordnet Search – 3.1). Šis išteklius buvo integruotas į mišriosios vertimo sistemos programą. Sprendimas realizuojamas užklausos į *Wordnet* bazę forma. Bendru atveju (žr. grafiškai pavaizduotą metodą 16 pav.) pirmoji užklausa yra nukreipiama į interneto išteklių, vadinamą morfologinių analizatoriumi, kuris paverčia žodžius jų žodyninėmis formomis (lemomis). *Wordnet* turi integruotą morfologinį analizatorių *Morphy*, todėl potencialių programos užklausų kiekis buvo sumažintas perpus.



16 pav. Siūlomo interneto išteklių naudojimo metodo sumaišymo tinklo karkaso ir vertimo sistemų žodžiams priskirti iliustracija

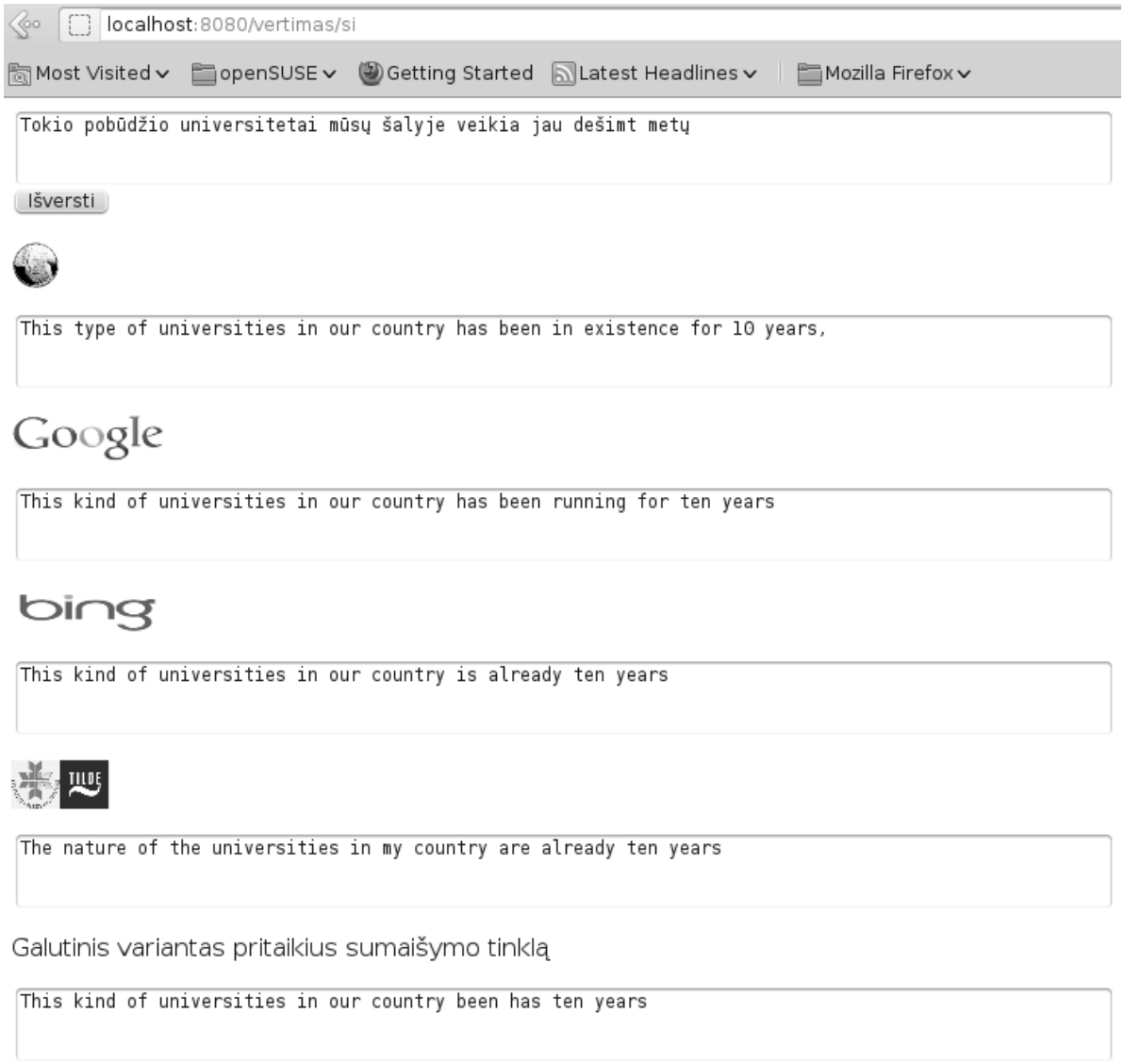
Toliau pateikiami pagrindiniai programos veikimo etapai:

1. **Vertimo sistemų rezultatų gavimas.** Pagal anksčiau pateiktas *HTTP* protokolo užklausas iš naudojamų internetinių vertimo sistemų (*Google Translate*, *Bing Translator*, *LKI ir Tilde IT*) gaunami lietuviško teksto vertimai. Naudojami scenarijai su *wget* komanda kartu su raktu –output-document, kas leidžia išsaugoti rezultatą į kietąjį diską. *Moses* vertimo sistemos rezultatas gaunamas kitu būdu. Šiai sistemai reikia pateikti iš anksto apdorotus duomenis, todėl iškviečiamas scenarijus, kuris įvesto teksto didžiąsias raides paverčia mažosiomis ir suskaido visą tekstą į simbolius. Visų keturių sistemų rezultatai yra įrašomi į eilutės tipo kintamuosius.
2. **Vertimo sistemų rezultatų suskaidymas į žodžius ir simbolius.** Šio etapo metu kviečiama funkcija *isskaidyti_i_simb*, kuri kiekvieną vertimo variantą išskaido į žodžius ir skyrybos ženklus. Kiekvienas rezultatas yra įrašomas į vektoriaus tipo kintamąjį.
3. **Žodžių priskyrimas.** Svarbiausias etapas, kurio metu pasirinkto karkaso (jį galima keisti) žodžiai priskiriami visų kitų sistemų vertimų žodžiams. Visų pirma, į globalųjį *Map* struktūros kintamąjį *sumaisymo_tinklas* yra įrašomi karkaso žodžiai ir jų pradinės pozicijos. Toliau kiekvienai sistemai kviečiama funkcija *priskirti_zodzius*. Atliekamas pradinis priskyrimas, kurio metu kiekvienas karkaso žodis sutikrinamas su kiekvienu tam tikros sistemos vertimo žodžiu. Žodžiai bus priskirti vienas kitam tik tuo atveju, jei jie visiškai sutaps. Priskyrimo rezultatai yra saugomi *Map* tipo struktūroje *vertimas_map*. Pavyzdyje, pateiktame 16 pav., pirmasis vertimo sistemos žodis sutapo su antruoju karkaso žodžiu, penktasis – su trečiuoju ir t.t. Kai pradinis priskyrimas baigtas, naudojamas siūlomas autoriaus metodas. Nepriskirti karkaso žodžiai (*karkasas_copy* struktūra programoje) kreipiantis į *Wordnet* paverčiami žodyninėmis žodžių formomis (lemomis). Sumaišymo tinklo potencialūs žodžiai-atitikmenys taip pat paduodami į *Wordnet* bazę. Toliau yra išsaugomi kiek nagrinėjamo karkaso žodžio (eilutė *page*, nes išsaugomas visas tinklalapis), tiek ir vertimo sistemos žodžio (*vertimo_zodziai* struktūra) sinonimai. Jei konkretaus karkaso žodžio vienas iš sinonimų sutapo su konkretaus sistemos žodžio sinonimu, šie žodžiai yra priskiriami vienas kitam, toliau nėra nagrinėjami ir programa pereina prie kito karkaso žodžio. Priskyrimas baigiasi, kai išnagrinėti visi karkaso žodžiai.
4. **Nepriskirtų žodžių įrašymas į tinkamas vietas / žodžių tvarkos nustatymas.** Po priskyrimo dažniausiai lieka žodžiai, kurie nerado savo atitikmens tarp karkaso žodžių. Kaip jau buvo minėta šio etapo uždaviniams spręsti buvo panaudotas minimalaus atstumo metodas. Jeigu tam tikros sistemos vertimas yra ilgesnis (daugiau žodžių) už karkaso vertimą ir sistemos žodžiai jau užima visas galimas pozicijas, karkasas yra praplečiamas. Į atsiradusią laisvą vietą yra įrašomas nagrinėjamas vertimo sistemos žodis, o karkasas toje pozicijoje turės tuščią eilutę.
5. **Sumaišymo tinklo žodžių tikėtinumo skaičiavimas / rezultatų išvedimas.** Šiame etape skaičiuojamos sumaišymo tinklo žodžių pasirodymo galutiniame variante tikimybės. Žodis bus

galutiniame variante, jeigu jis tam tikroje pozicijoje pasirodė bent dviejuose sumaišymo tinklo vertimų variantuose. Priešingu atveju imamas karkaso žodis (pavyzdžiui, jeigu pirmoje pozicijoje keturiuose vertimo variantuose visi žodžiai skirtingi).

Papildomos problemos, kurios kilo ir buvo išspręstos kuriant integruotą vertimo sistemą:

- lietuviškų raidžių konvertacija į/iš *HTML* kalbos kodus – šį darbą programoje atlieka funkcijos *ltConvert* ir *utf8Convert*;
- vertimo varianto skaidymas į žodžius ir simbolius – šį darbą programoje atlieka funkcija *isskaidyti_i_simb*;
- išorinių scenarijų kvietimas iš pagrindinės programos, jų įvesties ir išvesties apdorojimas (daugiausia pirmojo etapo metu).



17 pav. Kelių automatinio vertimo sistemų integracijos programos grafinė naudotojo sąsaja

Sukurta mišrioji vertimo sistema buvo įvertinta pateikus jai tuos pačius dvidešimt sakinių, kuriais remiantis buvo įvertintos visos pavienės vertimo sistemos (žr. 9 lentelę).

9 lentelė. Sukurtos mišriosios sistemos variantų įvertinimas pagal BLEU metriką remiantis dvidešimt sistemų išverstų sakinių

	BLEU-1	BLEU-2	BLEU-3	BLEU-4	BLEU-5	BLEU-6	BLEU-7	BLEU-8	BLEU-9
Mišrioji vertimo sistema 1 (karkasas - *Bing Translator*)	0.5488	0.3556	0.2385	**0.1637**	0.1112	0.0752	0.0527	0.0372	0.0264
Mišrioji vertimo sistema 2 (karkasas – *Google Translate*)	0.5296	0.3187	0.2017	**0.1297**	0.0824	0.0572	0.0401	0.0294	0.0214

Buvo įvertinti du mišriosios vertimo sistemos variantai – vienos karkasas yra *Bing Translator*, kitos – *Google Translate*. Palyginus šiuos duomenis su duomenimis, gautais 8 lentelėje, galima padaryti šias išvadas:

- pagrindinis rezultatas (*BLEU-4*) yra truputį mažesnis už atitinkamų karkasų, kuriais remiasi mišriosios sistemos, įvertinimus (16,75 % prieš 16,37 % ir 13,65 % prieš 12,97 % atitinkamai), tačiau pranoksta dviejų pavienių sistemų (*Tilde ir LKI* ir *Moses*) rezultatus (10,18 % ir 9,92 %);
- pirmosios mišriosios sistemos varianto atveju vertimas pagerėjo tik atsižvelgiant į pavienių žodžių vertimo metriką *BLEU-1* (54,88 % prieš 54,07 %);
- antrosios mišriosios sistemos varianto atveju vertimas pagerėjo keliais atvejais: unigramų atveju vertimas tikslesnis 0,39 %, tačiau labiausiai vertimas pagerėjo nagrinėjant ilgas žodžių sekas – *BLEU-8* tapo lygus 2,94 % (buvo 0 %), *BLEU-9* tapo lygus 2,14 % (buvo 0 %).

Išvados

Šiame baigiamojo magistro darbe nagrinėjamos pagrindinės automatinio vertimo problemos ir egzistuojančios vertimo sistemos tam, kad būtų galima pasiūlyti mišriosios automatinio vertimo sistemos variantą, kuris pateiktų tikslesnį vertimą iš lietuvių į anglų kalbą. Šiam tikslui pasiekti atlikti darbai ir gauti rezultatai, kurie pateikiami žemiau:

1. Pateikta teorinės literatūros apžvalga ir atlikta egzistuojančių automatinio vertimo sistemų analizė, nustatytas šių sistemų efektyvumas. Detaliai išnagrinėti statistinių vertimo sistemų komponentai, jų sąryšis ir jų įtaka galutiniam vertimui, apibūdinti pagrindiniai automatinio vertimo (statistinis ir tradicinis – taisyklių) metodai, išvardyti kiekvieno metodo privalumai ir trūkumai. Pristatyta automatinio vertimo įvertinimo metrika *BLEU*.
2. Atlikta lietuvių ir anglų kalbų lingvistinė lyginamoji analizė. Apibrėžtos pagrindinės automatinio vertimo problemos, susijusios su vertimu iš anglų į lietuvių ir iš lietuvių į anglų kalbas. Parodyta automatinio vertimo iš lietuvių į anglų kalbą svarba globalizacijos metu.
3. Išnagrinėtas programinis įrankis *Moses*. Atlikti tyrimai ir bandymai leido sudaryti visiškai veikiantį būsimos mišriosios vertimo sistemos pagrindinį komponentą – vertimo sistemą *Moses*. Atliktas šios automatinio vertimo sistemos palyginimas pagal *BLEU* metriką su egzistuojančiomis vertimo sistemomis. Sistema pasiekė 9,92 % įvertinimą pagal *BLEU-4* metriką, beveik pasiekdama nežymiai geriau pasirodžiusios *Tilde IT ir LKI* vertimo sistemos rezultatą (10,18 %).
4. Parodyti kelių automatinio vertimo sistemų integracijos privalumai ir perspektyvumas argumentuojant darbo temos pasirinkimą; detaliai išnagrinėti sumaišymo tinklai ir jų realizavimo būdai. Išnagrinėtos kelių automatinio vertimo sistemų integracijos metodikos ir architektūros, paremtos šiomis metodikomis.
5. Atliktas pirmasis Lietuvoje sėkmingas bandymas sukurti mišriąją vertimo sistemą. Ši sistema buvo paleista kaip interneto paslauga ir yra prieinama adresu http://84.32.228.222:8080/vertimas/si. Aprašytas programos veikimo principas. Sistemos rezultatai palyginti su pavienių sistemų rezultatais. Mišriosios sistemos variantas su *Bing Translator* karkasu gavo 16,37 % *BLEU-4* įvertinimą ir šiuo atžvilgiu nežymiai nusileido pavienei *Bing Translator* vertimo sistemai (16,75 %), tačiau sukurtos sistemos pavienių žodžių tikslumas (54,88 %) yra didesnis palyginus su paviene *Bing Translator* sistema. Antrasis mišiosios sistemos variantas (karkasas – *Google Translate*) taip pat pranoko pavienį karkasą pagal *BLEU-1* įvertinimą (52,96 % prieš 52,57 % ir nusileido karkasui pagal *BLEU-4* rezultatą (12,97 % prieš 13,65 %). Tačiau labiausiai mišrioji sistema pranoko karkasą pagal aukštos eilės *BLEU* metrikos rodiklius. *BLEU-8* tapo lygus 2,94 % (buvo 0 %), *BLEU-9* – 2,14 % (buvo 0 %). Abu mišriosios

sistemos variantai pasirodė geriau nei dvi pavienės vertimo sistemos (*Tilde ir LKI* ir *Moses*), kurių *BLEU-4* rezultatai yra 10,18 % ir 9,92 % atitinkamai.

Gauti rezultatai leidžia pateikti šias išvadas:

1. Iškelta hipotezė, kad mišrioji vertimo sistema turėtų pateikti geresnį vertimą nei pavienės sistemos, dalinai pasitvirtino. Abu mišriosios sistemos variantai parodė aukštesnį metrikos *BLEU-1* rezultatą. Taigi pavieniai žodžiai yra verčiami tiksliau. Aukštesnio lygio *BLEU* metrikų rezultato dar nepavyko pasiekti, todėl kad dar ne iki galo išspręsta žodžių tvarkos problema.
2. Įsitikinta ir pademonstruota, kad sumaišymo tinklo kūrimo problema yra sunkiai sprendžiama. Parodyta, kad pasiūlyta naujoviška koncepcija – naudoti interneto išteklius sumaišymo tinklui sudaryti – pagerina karkaso siūlomą vertimo variantą. Programai veikiant be šio metodo atsirastų daug nepriskirtų vienas kitam žodžių, o tai smarkiai sumažintų vertimo tikslumą. Netgi, jeigu galutinis variantas nėra tikslus, sumaišymo tinklo taikymas automatinio vertimo sistemai sukurti palengvina naudotojo darbą, nes leidžia pamatyti visus įmanomus vertimo variantus ir visi šie variantai pasirodo vienu metu. Taip pat sistemos palengvina profesionalių vertėjų postredagavimo darbą, nes suteikia jiems pasirinkimo galimybę ir tai padeda išrinkti geriausią vertimo variantą.
3. Atlikus tam tikrus programinio kodo pakeitimus, sukurta mišrioji vertimo sistema gali būti pritaikyta bet kurių kitų dviejų kalbų porai. Taip pat yra lengva pakeisti sistemos karkasą bei praplėsti sumaišymo tinklą naujomis vertimo sistemomis.
4. Atliktas lietuvių ir anglų kalbų lingvistinis palyginimas leidžia nustatyti nepakankamo vertimo tikslumo priežastis. Galima teigti, kad vertimo sistemų kūrimas priklauso nuo kalbų poros ir šių kalbų išteklių nevienodumo (skurdžių duomenų problema). Lietuvių ir anglų kalboms akivaizdžiai trūksta kokybiškų didelės apimties tekstynų, kurie padėtų pasiekti geresnį statistinį vertimą šiai kalbų porai.
5. Vienintelė Lietuvoje sukurta vertimo sistema verčianti iš lietuvių į anglų kalbą, kuri nors ir yra paremta *Moses* vertimo sistema, kaip parodyta skyriuje 2.1.1, pateikia neteisingą ir klaidinantį kai kurių sakinių vertimą. Tuos pačius sakinius autoriaus sukurta *Moses* vertimo sistema sugeba išversti pakankamai gerai, todėl galima teigti, kad ši pavienė *Moses* sistema gali sėkmingai konkuruoti su *LKI ir Tilde* siūloma sistema.

Kuriant vertimo sistemą buvo išspręsti papildomi programavimo uždaviniai: sužinoti ir panaudoti egzistuojančių internetinių vertimo sistemų užklausų eilutes *HTTP* protokolu; nustatyta galutinio vertimo varianto žodžių tvarka pagal autoriaus pasiūlytą minimalaus atstumo metodą; lietuviškos raides konvertuotos į atitinkamus *HTML* kalbos kodus. Galima paminėti, kad dėl jų sudėtingumo automatinio

vertimo programos paprastai yra kolektyvinio darbo rezultatas, bet ši programa sukurta vien autoriaus pastangomis.

Sprendžiamos problemos ir gauti rezultatai rodo naujoviškumą, praktinę ir teorinę darbo vertę. Darbo rezultatai gali būti naudojami informatikos specialistų rašant panašaus pobūdžio vertimo sistemų programas, o taip pat naudotojų (įskaitant profesionalius vertėjus), norinčių iškart sužinoti visus įmanomus teksto vertimo variantus.

Tolimesni darbai turi būti nukreipti į automatinio vertimo tikslumo didinimą, kurį galima pasiekti naudojant šiuos būdus:

- papildomų tekstynų ruošimas ir naudojimas *Moses* vertimo sistemoje;
- informacijos paieškos tinklalapiuose laiko mažinimas (pavyzdžiui, naudojant gijas);
- papildomų tinklalapių naudojimas sinonimų paieškai;
- žodžių tvarkos problemos sprendimas naudojant įvairius statistinius metodus;
- vienos vertimo sistemos žodžių priskyrimas ne tik karkasui bet ir kitoms sistemoms.

Literatūra

1. Albrektas, T. 2010. Apie statistinį mašininį vertimą [interaktyvus]. [žiūrėta 2011-10-25]. Prieiga per internetą: http://blog.lituanika.lt/.
2. Bhavnani, R. 2010. Google Translator [interaktyvus]. [žiūrėta 2012-01-15]. Prieiga per internetą: http://www.codeproject.com/Articles/12711/Google-Translator/.
3. Bing Translator [interaktyvus]. [žiūrėta 2012-05-20]. Prieiga per internetą: http://www.microsofttranslator.com/.
4. Brown, P., Della-Pietra, S., Della-Pietra, V., Mercer, R. 1993. The mathematics of machine translation: parameter estimation. *Computational Linguistics*, nr. 19(2), p. 263-311.
5. Callison-Burch, C, Koehn, P., Monz, C., Schroeder, J.. 2009. Findings of the 2009 Workshop on Statistical Machine Translation. Proceedings of the Fourth Workshop on Statistical Machine Translation. Athens, p. 1–28.
6. Cancedda, N., Dymetman, M., Foster, G., Goutte, C. 2009. A statistical machine translation primer. *Learning Machine Translation*. p. 1-37.
7. Carl, M., Way, A., Schaler, R. 2002. Toward a Hybrid Integrated Translation Environment // 5th Conference of the Association for Machine Translation in the Americas: Proceedings [Tiburon, CA, USA, October 8-12 2002]. Tiburon, p. 11-20.
8. Chen, Y., Eisele, A. 2010. Rule-based translation with statistical phrase-based post-editing // Seventh International Conference on Language Resources and Evaluation: Proceedings // [Valletta, Malta, May 19-21 2010].
9. Dugast, L., Senellart, J., Koehn, P. 2007. Statistical post-editing on SYSTRAN's rule-based translation system // // WMT07: Proceedings // [Prague, Czech Republic, June 23 2007]. Prague, p. 220-223.
10. Eisele, A. 2007. Hybrid machine translation: combining rule-based and statistical systems // The First Machine Translation Marathon // [Edinburgh, Great Britain, April 17 2007].
11. Eisele, A., Federmann, C., Saint-Armand, H. et al. 2008. Using Moses to Integrate Multiple Rule-Based Machine Translation Engines into a Hybrid System// the Third Workshop on Statistical Machine Translation: Proceedings // [Columbus, Ohio, USA, June 19 2008], p. 179-182.
12. Google vertėjas [interaktyvus]. [žiūrėta 2012-05-05]. Prieiga per internetą: http://translate.google.lt/.
13. KLC [interaktyvus]. [žiūrėta 2012-04-30]. Prieiga per internetą: http://tekstynas.vdu.lt/page.xhtml?id=tools.
14. Knight, K. 1999. Decoding complexity in word-replacement translation models. *Computational Linguistics,* nr. 25(4), p. 607–615.

15. Koehn, P. 2010. *Statistical Machine Translation*, Cambridge University Press. 433 p. ISBN 978-0-521-87415-1.
16. Koehn, P., Hoang, H., Birch, A. et al. 2007. Moses: Open Source Toolkit for Statistical Machine Translation // Annual Meeting of the Association for Computational Linguistics // [Prague, Czech Republic, June 23 2007]. Prague, p. 177-180.
17. Laukaitis, A., Vasilecas, O. 2007. Asymmetric hybrid machine translation for languages with scarce resources // Computational Linguistics and Intelligent Text Processing: 8th International Conference: Proceedings // [Mexico City, Mexico, February 18-24, 2007]. Berlin: Springer. ISSN 0302-9743. Vol. 4394, p. 397-408.
18. Lietuvos Respublikos terminų bankas [interaktyvus]. [žiūrėta 2012-05-10]. Prieiga per internetą: http://terminai.vlkk.lt/.
19. Marin, I. 2010. *Techninio teksto kompiuterinio vertimo programos besimokantiems ir dirbantiems statybos srityje*: baigiamasis bakalauro darbas: fiziniai mokslai: informatika / Igor Marin: Vilniaus Gedimino technikos universitetas. Vilnius.
20. Mašininis vertimas [interaktyvus]. [žiūrėta 2012-05-13]. Prieiga per internetą: http://mvlab.lki.lt/.
21. Matusov, E., Leusch, G., Ney, H. 2009. Learning To Combine Machine Translation Systems. *Learning Machine Translation*. p. 257-276.
22. Moses [interaktyvus]. [žiūrėta 2012-04-21]. Prieiga per internetą: http://www.statmt.org/moses/.
23. Norvig, P. 2007. Theorizing from data: avoiding the capital mistake [interaktyvus]. [žiūrėta 2010-12-11]. Prieiga per internetą: http://www.youtube.com/watch?v=nU8DcBF-qo4.
24. Peng, F. 2001. The Sparse Data Problem in Statistical Language Modeling and Unsupervised Word Segmentation: PhD thesis [interaktyvus]. [žiūrėta 2012-03-27]. Prieiga per internetą: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.24.5265&rep=rep1&type=pdf.
25. Rimkutė, E., Kovalevskaitė, J. 2007. Mašininis vertimas – greitoji pagalba globalėjančiam pasauliui. *Gimtoji kalba*: *bendrinės kalbos žurnalas*, nr. 9, p. 3-11.
26. Rosinaitė, V. 2010. *Constructing the concept of career in Lithuania*: daktaro disertacija: socialiniai mokslai: sociologija / Vikinta Rosinaitė: Vilniaus universitetas. Vilnius.
27. Rosti, A.-V. I., Zhang, B., Matsoukas, S., Schwartz, R. 2008. Incremental hypothesis alignment for building confusion networks with application to machine translation system combination // the Third Workshop on Statistical Machine Translation: Proceedings // [Columbus, Ohio, USA, June 19 2008], p. 183-186.
28. Simard, M., Ueffing, N., Isabelle, P., Kuhn, R. 2007. Rule-based translation with statistical phrase-based post-editing // WMT07: Proceedings // [Prague, Czech Republic, June 23 2007]. Prague, p. 203-206.

29. Teksto vertimas [interaktyvus]. [žiūrėta 2012-04-25]. Prieiga per internetą: http://vertimas.vdu.lt/twsas/.
30. The JRC-Acquis Multilingual Parallel Corpus [interaktyvus]. [žiūrėta 2011-01-11]. Prieiga per internetą: http://langtech.jrc.it/JRC-Acquis.html
31. Tilde IT. 2012. Tildės Biuras 2012 – naujas žingsnis vertimų srityje [interaktyvus]. [žiūrėta 2012-05-16]. Prieiga per internetą: http://vz.lt/?PublicationId=633735e1-6b27-4566-b37b-2635a0b7b7a9.
32. Wordnet Search – 3.1 [interaktyvus]. [žiūrėta 2012-05-05]. Prieiga per internetą: http://wordnetweb.princeton.edu/perl/webwn
33. Андреев, А. 2007. Машинный перевод: правила против статистики. *CIO*, №8.

Priedai. Kelių automatinio vertimo sistemų integracijos programos kodas *Java* kalba

```
import java.io.BufferedInputStream;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.URL;
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.Map;
import java.util.Scanner;
import java.util.Set;
import java.util.Vector;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class si
 */
@WebServlet(description = "sistemu integracija", urlPatterns = { "/si" })
public class si extends HttpServlet {
	private static final long serialVersionUID = 1L;
    /**
     * @see HttpServlet#HttpServlet()
     */
	public static Map<Integer, Map> sumaisymo_tinklas =
			new LinkedHashMap<Integer, Map>();
	public static Map<Integer,String> karkasas_map =
			new LinkedHashMap<Integer, String>();
	//Sistemos numeris
	public static int sistema = 0;
    public si() {
        super();
    }
	/**
	 * @see HttpServlet#doGet(HttpServletRequest request,
	 * HttpServletResponse response)
	 */
	protected void doGet(HttpServletRequest request,
			HttpServletResponse response) throws ServletException, IOException {
		PrintWriter out = response.getWriter();
		String iveskite = "Įveskite tekstą, kurį norite išversti...";
		out.println("<!DOCTYPE HTML PUBLIC"+
		"\"-//W3C//DTD HTML 4.01 Transitional//EN\""+
		"\"http://www.w3.org/TR/1999/REC-html401-19991224/loose.dtd\">" +
		"<HTML><HEAD>"+
		"<TITLE>Automatinio vertimo sistemu integracija</TITLE>"+
		"<META HTTP-EQUIV=\"Content-Type\""+
		"CONTENT=\"text/html; charset=utf-8\">"+
		"</HEAD><BODY>"+
		"<form method=\"post\" action=\"\">"+
		"<textarea onFocus=\"this.value=''; this.onfocus=null;\""+
```

```
            "name=\"ivestis\"rows=\"3\" cols=\"100\" >"+
            ltConvert(iveskite)+
            "</textarea><br>"+"<input type=\"submit\" value=\"" +
            ltConvert("Išversti")+"\" onclick=\"javascript: "+
            "isvestis_cn.value = ivestis.value = ''\""+
            "/></form><p>"+
            "<img src=\"http://www.statmt.org/moses/img/coin-tiny.jpg\""+
            "alt=\"Moses\" width=\"32\" height=\"32\" /> </p>"+
            "<textarea name=\"isvestis_moses\" cols=\"100\" rows=\"2\">"+
            "</textarea>"+
            "</form><p>"+
            "<img src=\"http://upload.wikimedia.org/wikipedia/commons/"+
            "thumb/2/20/Google-Logo.svg/220px-Google-Logo.svg.png\""+
            "alt=\"Google\" width=\"100\" height=\"32\" /> </p>"+
            "<textarea name=\"isvestis_google\" cols=\"100\" rows=\"2\">"+
            "</textarea>"+
            "</form><p>"+
            "<img src=\"http://upload.wikimedia.org/wikipedia/commons/"+
            "thumb/e/e9/Bing_logo.svg/166px-Bing_logo.svg.png\""+
            "alt=\"Bing\" width=\"100\" height=\"32\" /> </p>"+
            "<textarea name=\"isvestis_bing\" cols=\"100\" rows=\"2\">"+
            "</textarea><p>"+
            "<img src=\"http://mvlab.lki.lt/img/lki.png\""+
            "alt=\"LKI\" width=\"32\" height=\"32\" />" +
            "<img src=\"http://mvlab.lki.lt/img/logo.png\""+
            "alt=\"Tilde\" width=\"32\" height=\"32\" /> </p>"+
            "<textarea name=\"isvestis_tilde\" cols=\"100\" rows=\"2\">"+
            "</textarea><p>"+
            ltConvert("Galutinis variantas pritaikius sumaišymo tinklą")+
            "</p><textarea name=\"isvestis_cn\" cols=\"100\" rows=\"2\">"+
            "</textarea><p>"+
            "</BODY></HTML>");
    }
    /**
     * @see HttpServlet#doPost(HttpServletRequest request,
     * HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request,
                    HttpServletResponse response) throws ServletException, IOException {
            String ivestis = utf8Convert(request.getParameter("ivestis"));
            if (!ivestis.substring(ivestis.length()-1,
                            ivestis.length()).contains("."))
                    ivestis+=" .";

            //Google vertimas
       try {
            BufferedWriter out = new BufferedWriter(
                            new FileWriter("/home/igor/google_input.txt"));
            out.write("http://translate.google.com/?text=" +
                            ivestis+"&langpair=lt|en");
            out.close();
            Process p = Runtime.getRuntime().exec("sh /home/igor/google.sh");
                    p.waitFor();
                    p.destroy();
       }
            catch (InterruptedException e1) {
                    e1.printStackTrace();
            }
            catch (IOException e)           {
            System.out.println("Exception");
            }
      //Google vertimo nuskaitymas is html failo
```

```
                String result_google = "";
                try {
                  BufferedReader br = new BufferedReader(
                                  new FileReader("/home/igor/google.txt"));
                  String line;
                  while((line = br.readLine()) != null) {
                            if(line.contains("='#fff")){
                             int index = line.indexOf("='#fff");
                             result_google = line.substring(index+9);
                             result_google = result_google.substring(0,
                                              result_google.indexOf("<"));
                             }
                  }
            }
                  catch ( IOException ex ) {
                      System.out.println("Failas negali buti atidarytas");}

//Bing vertimas
  try {
        BufferedWriter out = new BufferedWriter(
                          new FileWriter("/home/igor/bing_input.txt"));
        String appID = "F1B50AB0743B541AA8C07089042D7B57E9B28D25";
        out.write("http://api.microsofttranslator.com/V2/Ajax.svc/"+
        "Translate?appId="+appID+"&from=lt&to=en&text="+ivestis);
        out.close();
        }
        catch (IOException e)
        {
        System.out.println("Exception");
  }
//Bing vertimo nuskaitymas is gauto failo
        String result_bing = "";
        try {
                  Process p = Runtime.getRuntime().exec("sh /home/igor/bing.sh");
                  p.waitFor();
                  p.destroy();
                  File file = new File("/home/igor/bing.txt");
                  Scanner scanner = new Scanner(file);
                  while(scanner.hasNextLine()) {
                 result_bing+=scanner.nextLine();
              }
                  } catch (InterruptedException e1) {
                  e1.printStackTrace();}
                  catch (IOException e)  {System.out.println("Exception");}

        //Moses vertimas
        String result_moses = "", ivestis_paruosta = "";
         try {
                   BufferedWriter out = new BufferedWriter(
                                  new FileWriter("/home/igor/moses_input.txt"));
                  out.write(ivestis);
                  out.close();
             String line;
                   String[] cmd1 = {"sh", "/home/igor/moses_paruosimas.sh"};
                   Process p = Runtime.getRuntime().exec(cmd1);
                   BufferedReader bri = new BufferedReader
                   (new InputStreamReader(p.getInputStream()));
                   while ((line = bri.readLine()) != null) {
                             ivestis_paruosta+= line;
                   }
                   if (ivestis_paruosta.substring(ivestis_paruosta.length()-1,
                                        ivestis_paruosta.length()).contains("."))
```

```
                                ivestis_paruosta = ivestis_paruosta.replace(".", " .");
            bri.close();
            p.waitFor();
            p.destroy();
     String[] cmd = {"sh", "/home/igor/moses.sh",
             ivestis_paruosta.toLowerCase()};
     p = Runtime.getRuntime().exec(cmd);
     bri = new BufferedReader
       (new InputStreamReader(p.getInputStream()));
     while ((line = bri.readLine()) != null) {
            result_moses+= line;
     }
      bri.close();
      p.waitFor();
             p.destroy();
    }
    catch (Exception err) {
      err.printStackTrace();
    }

//LKI ir Tilde sistemos vertimas
    try {
            BufferedWriter out = new BufferedWriter(
                                new FileWriter("/home/igor/tilde_input.txt"));
            out.write("http://mvlab.lki.lt/getTranslation.aspx?&text=" +
                                ivestis);
            out.close();
            Process p = Runtime.getRuntime().exec(
                                "sh /home/igor/tilde.sh");
                        p.waitFor();
                        p.destroy();
            } catch (InterruptedException e1) {
                        e1.printStackTrace();}
            catch (IOException e)  {System.out.println("Exception");}

//LKI ir Tilde sistemos vertimo nuskaitymas is gauto failo
    Runtime.getRuntime().exec("sh /home/igor/tilde.sh");
            String result_tilde = "";
            try {
                        File file = new File("/home/igor/tilde.txt");
                        Scanner scanner = new Scanner(file);
                        while(scanner.hasNextLine()) {
                      result_tilde+=scanner.nextLine();
                    }
                  }
                  catch ( IOException ex ) {
                    System.out.println("Failas negali buti atidarytas");
                  }
result_google = result_google.replaceAll("&#39;","'");
result_bing =
result_bing.substring(2, result_bing.length()-1);
result_tilde =
                        result_tilde.substring(result_tilde.indexOf(":")+3,
                          result_tilde.indexOf(", \"wordcount\":")-1);

Vector<String> bing = isskaidyti_i_simb(result_bing + " ");
Vector<String> google = isskaidyti_i_simb(result_google + " ");
Vector<String> moses = isskaidyti_i_simb(result_moses + " ");
Vector<String> tilde = isskaidyti_i_simb(result_tilde + " ");
//Sumaisymo tinklo karkaso pasirinkimas
Vector<String> karkasas = bing;
for (int i = 0; i < karkasas.size();i++)
```

```
karkasas_map.put(i, karkasas.get(i));
sumaisymo_tinklas.put(0,karkasas_map);
priskirti_zodzius(google, karkasas_map);
karkasas_map = sumaisymo_tinklas.get(0);
priskirti_zodzius(moses, karkasas_map);
karkasas_map = sumaisymo_tinklas.get(0);
priskirti_zodzius(tilde, karkasas_map);

String galutinis_vertimas = "";
int vertimo_ilgis = -1;

for (int k = 0; k < sumaisymo_tinklas.size(); k++){
        if (sumaisymo_tinklas.get(k).size()> vertimo_ilgis)
                vertimo_ilgis = sumaisymo_tinklas.get(k).size();
}

for (int k = 0; k < vertimo_ilgis; k++) {
        String[][] zodziu_variantai =
                        new String[sumaisymo_tinklas.size()][2];
        for (int index = 0; index < sumaisymo_tinklas.size(); index++)
                zodziu_variantai[index][1] = "0";
        for (int sys = 0; sys < sumaisymo_tinklas.size(); sys++) {

        zodziu_variantai[sys][0] =
                        (String)sumaisymo_tinklas.get(sys).get(k);
        }

for (int l = 0; l < sumaisymo_tinklas.size(); l++){
        int kartai = 0;
        for (int m = 0; m < sumaisymo_tinklas.size(); m++){
                if (l == m) continue;
                if (zodziu_variantai[l][0] != null &&
                        !zodziu_variantai[l][0].isEmpty() &&
                        zodziu_variantai[m][0] != null &&
                        !zodziu_variantai[m][0].isEmpty() &&
                        zodziu_variantai[l][0].
                        toLowerCase().equals(
                        zodziu_variantai[m][0].toLowerCase())) {
                                kartai++;
                                zodziu_variantai[l][1] = Integer.toString(kartai);
                }
        }
}
int zodis = 0;
for (int i = 0; i < zodziu_variantai.length; i++) {
        if (zodziu_variantai[i][1] != null &&
                        Integer.parseInt(zodziu_variantai[i][1]) >
                        Integer.parseInt(zodziu_variantai[zodis][1]))
                zodis = i;
}
if (zodziu_variantai[zodis][0] != null &&
                !zodziu_variantai[zodis][0].equals("")) {
        if (zodziu_variantai[zodis][0].equals(",") ||
                zodziu_variantai[zodis][0].equals("."))
                galutinis_vertimas = galutinis_vertimas.substring
                                                (0,galutinis_vertimas.length()-1);
        galutinis_vertimas+=zodziu_variantai[zodis][0]+" ";
        }
}

Map<Integer,String> zodziai =
                new LinkedHashMap<Integer, String>();
```

```
            sistema = 0;
            sumaisymo_tinklas =
                        new LinkedHashMap<Integer, Map>();
            karkasas_map = new LinkedHashMap<Integer, String>();

            PrintWriter out = response.getWriter();
            out.println("<!DOCTYPE HTML PUBLIC"+
            "\"-//W3C//DTD HTML 4.01 Transitional//EN\""+
            "\"http://www.w3.org/TR/1999/REC-html401-19991224/loose.dtd\">" +
            "<HTML><HEAD>"+
            "<TITLE>Automatinio vertimo sistemu integracija</TITLE>"+
            "<META HTTP-EQUIV=\"Content-Type\""+
            "CONTENT=\"text/html; charset=utf-8\">"+
            "</HEAD><BODY>"+
            "<form method=\"post\" action=\"\">"+
            "<textarea name=\"ivestis\" cols=\"100\" rows=\"2\">"+
            ltConvert(ivestis)+
            "</textarea><br>"+"<input type=\"submit\" value=\"" +
            ltConvert("Išversti")+"\" onclick=\"javascript: " +
            "isvestis_cn.value = ivestis.value = ''\""+
            "/></form><p>"+
            "<img src=\"http://www.statmt.org/moses/img/coin-tiny.jpg\""+
            "alt=\"Moses\" width=\"32\" height=\"32\" /> </p>"+
            "<textarea name=\"isvestis_moses\" cols=\"100\" rows=\"2\">"+
            ltConvert(result_moses)+
            "</textarea><p>"+
            "<img src=\"http://upload.wikimedia.org/wikipedia/commons"+
            "/thumb/2/20/Google-Logo.svg/220px-Google-Logo.svg.png\""+
            "alt=\"Google\" width=\"100\" height=\"32\" /> </p>"+
            "<textarea name=\"isvestis_google\" cols=\"100\" rows=\"2\">"+
            ltConvert(result_google) +
            "</textarea><p>"+
            "<img src=\"http://upload.wikimedia.org/wikipedia/commons/"+
            "thumb/e/e9/Bing_logo.svg/166px-Bing_logo.svg.png\""+
            "alt=\"Bing\" width=\"100\" height=\"32\" /> </p>"+
            "<textarea name=\"isvestis_bing\" cols=\"100\" rows=\"2\">"+
            ltConvert(result_bing)+
            "</textarea><p>"+
            "<img src=\"http://mvlab.lki.lt/img/lki.png\""+
            "alt=\"LKI\" width=\"32\" height=\"32\" />" +
            "<img src=\"http://mvlab.lki.lt/img/logo.png\""+
            "alt=\"Tilde\" width=\"32\" height=\"32\" /> </p>"+
            "<textarea name=\"isvestis_tilde\" cols=\"100\" rows=\"2\">"+
            ltConvert(result_tilde)+
            "</textarea><p>"+
            ltConvert("Galutinis variantas pritaikius sumaišymo tinklą")+
            "</p><textarea name=\"isvestis_cn\" cols=\"100\" rows=\"2\">"+
            ltConvert(galutinis_vertimas.substring(0,
                        galutinis_vertimas.length()-1))+
            "</textarea><p>"+
            "</BODY></HTML>");
            System.gc();
}

public static final String utf8Convert(String utf8String) throws
java.io.UnsupportedEncodingException {
byte[] bytes = new byte[utf8String.length()];
for (int i = 0; i < utf8String.length(); i++) {
bytes[i] = (byte) utf8String.charAt(i);
}
return new String(bytes, "UTF-8");
```

```
}

//Funkcija lietuviskoms raidems konvertuoti i ju html kodus
public static final String ltConvert(String string) {
        String[][] lt_raides = {{"ą" , "&#261;"},{"č" , "&#269;"},
        {"ę" , "&#281;"},{"ė" , "&#279;"},{"į" , "&#303;"},
        {"š" , "&#353;"},{"ų" , "&#371;"},{"ū" , "&#363;"},
        {"ž" , "&#382;"},{"Ą" , "&#260;"},{"Č" , "&#268;"},
        {"Ę" , "&#280;"},{"Ė" , "&#278;"},{"Į" , "&#302;"},
        {"Š" , "&#352;"},{"Ų" , "&#370;"},{"Ū" , "&#362;"},
        {"Ž" , "&#381;"}};

         for (int r=0; r<lt_raides.length; r++) {
               string = string.replaceAll(lt_raides[r][0],
                        lt_raides[r][1]);
         }
return string;
}

//Funkcija vertimo eilutei isskaidyti i simbolius
public static final Vector<String> isskaidyti_i_simb(
                        String vertimas) {
                Vector<String> v = new Vector<String>();
                String[] lt_raides = {"ą","č","ę","ė","į","š","ų","ū","ž",
                                      "Ą","Č","Ę","Ė","Į","Š","Ų","Ū","Ž"};
                        int pradzia = 0, spec_simb = 0, raide_ar_sk = 0;

               for (int i=0; i<=vertimas.length(); ) {
                        raide_ar_sk = 0;
                        if (vertimas.length() == 1) {
                                i=2; continue;
                        }
                        pradzia = 0;
                        if (vertimas.substring(0,1).equals(" ")) {
                                vertimas = vertimas.substring(1);
                                spec_simb = 0;
                                continue;
                        }
                        if (spec_simb == 1 &&
                                vertimas.substring(i,i+1).equals(" ")) {
                                v.add(vertimas.substring(0,i-1));
                                v.add(vertimas.substring(i-1,i));
                                vertimas = vertimas.substring(i);
                                i = 0; spec_simb = 0; continue;
                        }
                        String a = vertimas.substring(i,i+1);
                        if (vertimas.substring(i,i+1).equals(" ")) {
                                        v.add(vertimas.substring(0,i));
                                        vertimas = vertimas.substring(i);
                                        spec_simb = 0;
                                        i = 0;
                                        continue;
                        }
                        else {

                        if (vertimas.substring(i,i+1).matches(
                                "^[a-zA-Z0-9]+$") || (
                                vertimas.substring(i,i+1).matches(
                                "^[a-zA-Z0-9]+$") && i > 0 &&
                                vertimas.substring(i-1,i).matches(
                                "^[a-zA-Z0-9]+$")) ||
                                (i+2 <= vertimas.length() &&
```

```
                                        vertimas.substring(i+1,i+2).matches(
                                        "^[a-zA-Z0-9]+$")))
                                        {
                                        raide_ar_sk = 1;
                                        if (spec_simb == 1) {
                                                v.add(vertimas.substring(0,i));
                                                vertimas = vertimas.substring(i);
                                                i = 0; spec_simb = 0;
                                        }
                                        i++;
                                }
                                if (raide_ar_sk == 1) continue;

                                for (int r=0; r<lt_raides.length; r++) {
                                        if (vertimas.substring(i,i+1).equals(
                                                lt_raides[r])) {
                                                raide_ar_sk = 1;
                                                pradzia = 1;
                                                if (spec_simb == 1) {
                                                        v.add(vertimas.substring(0,i));
                                                        vertimas = vertimas.substring(i);
                                                        i = -1; spec_simb = 0; break;
                                                }
                                                else break;
                                                }
                                        }
                                        if (pradzia == 1) {i++; continue;}

                                        if (raide_ar_sk == 1) {
                                                v.add(vertimas.substring(0,i));
                                                vertimas = vertimas.substring(i);
                                                i = 0; spec_simb = 0;
                                                raide_ar_sk = 0; continue;
                                        }
                                                if (spec_simb == 0)
                                                        spec_simb = 1;
                                                        i++;
                                }
                        }
                return v;
                }

//Funkcija vertimo eilutei isskaidyti i simbolius
public static final Map<Integer,String> priskirti_zodzius(
                Vector<String> vertimas,
                //Vector<String> karkasas)
                Map<Integer,String> karkasas)
        {
        ++sistema;
        Map<Integer,String> vertimas_copy =
                        new LinkedHashMap<Integer, String>();
        for (int i = 0; i < vertimas.size();i++)
                vertimas_copy.put(i, vertimas.get(i));

        Map<Integer,String> vertimas_map =
                        new LinkedHashMap<Integer, String>();

        Map<Integer,String> vertimas_map_new;
        Map<Integer,String> karkasas_map_new;
        Map<Integer,Vector> vertimo_zodziai =
                        new LinkedHashMap<Integer, Vector>();
```

```
        Map<Integer,String> karkasas_copy = new LinkedHashMap<Integer, String>();
                        for (int k = 0; k < karkasas.size();k++)
                                karkasas_copy.put(k, karkasas.get(k));

        for (int i = 0; i < karkasas.size(); i++) {
                if (karkasas.get(i) == null ||
                        karkasas.get(i).equals("")) {
                        karkasas_copy.remove(i);
                        continue;
                }
                for (int j = 0; j < vertimas.size(); j++) {
                        if (vertimas.get(j).toLowerCase().equals(
                                karkasas.get(i).toLowerCase())) {
                                vertimas_map.put(i, vertimas.get(j));
                                karkasas_copy.remove(i);
                                vertimas.remove(j);
                                break;
                        }
                }
        }

        Set s = karkasas_copy.entrySet();
Iterator it = s.iterator();
String line = "",
                page = "", page2 = "";
URL url;
DataInputStream dis;
if (s.size() > 0) {
        try {
        for (int j = 0; j < vertimas.size(); j++) {
                if (vertimas.get(j).equals("")) {
                        vertimas.remove(j); --j;
                                continue;
                }
                url = new URL(
                "http://wordnetweb.princeton.edu/perl/webwn?s="
                +vertimas.get(j));
                dis = new DataInputStream(
                new BufferedInputStream(url.openStream()));
                Scanner scanner = new Scanner(new DataInputStream(dis));
                scanner.useDelimiter("<div class=\"key\">Display");
                String a = scanner.next();
                if (a.contains(
                                "<h3>Your search did not return any results.</h3>")
                ||
                a.contains("<h3>Sorry, your search can only contain letters"))
                                continue;
                page2 = scanner.next();
                Vector<String> zodis = new Vector<String>();

                        while (page2.contains(";s=")) {
                                int ind1 = page2.indexOf("\">");
                                int ind2 = page2.indexOf(";s=");
                                        if (!page2.contains("\">") || ind1 < ind2)
                                                page2 = page2.substring(page2.indexOf(";s="));
                                        else
                                                page2 = page2.substring(page2.indexOf(";s="),
                                                page2.lastIndexOf("\">"));
                                   if (!page2.contains("\">"))
                                                line = page2;
                                        else
                                                line = page2.substring(
```

```
                                                page2.indexOf(";s="),
                                                page2.indexOf("\">")+1);
                                int neideti = 0;
                                if (line.contains(
                                                vertimas.get(j).toLowerCase()+"&amp;i=")
                                                ||
                                        line.contains("+"))
                                                neideti = 1;
                                else {
                                for (int k = 0; k < zodis.size(); k++) {
                                if (
                                zodis.get(k).toLowerCase().equals(line.toLowerCase())) {
                                        neideti = 1;
                                        break;
                                                }
                                        }
                                }
                                if (neideti == 0)
                                zodis.add(line);
                                        if (page2.contains("\">"))
                                                page2 = page2.substring(page2.indexOf("\">")+2);
                                        else break;
                        }
                        vertimo_zodziai.put(j, zodis);
                        }
        }
        catch (IOException e) {
                        e.printStackTrace();
                }
}

while(it.hasNext()) {
        Set zodziu_variantai = vertimo_zodziai.entrySet();
        Iterator it_zv = zodziu_variantai.iterator();
        Map.Entry m =(Map.Entry)it.next();
        int surasta = 0;
        try {
                url = new URL(
                "http://wordnetweb.princeton.edu/perl/webwn?s="
                +(String)m.getValue());
                dis = new DataInputStream(
                new BufferedInputStream(url.openStream()));
                Scanner scanner = new Scanner(new DataInputStream(dis));
                scanner.useDelimiter("<div class=\"key\">Display");
                String a = scanner.next();
                if (a.contains(
                                "<h3>Your search did not return any results.</h3>")
                ||
                a.contains("<h3>Sorry, your search can only contain letters"))
                                continue;
                page = scanner.next();
                while(it_zv.hasNext()) {
                        if (surasta == 1) break;
                        Map.Entry mzv =(Map.Entry)it_zv.next();
                        for (int k = 0; k < ((Vector)mzv.getValue()).size(); k++) {
                                String z =
                                ((String)
                                ((Vector)mzv.getValue()).get(k)).toLowerCase();
                                if (page.contains(z))      {
                            vertimas_map.put(
                                                (Integer)m.getKey(),
                                                vertimas.get((Integer)mzv.getKey()));
```

```
                                    vertimas.set((Integer)mzv.getKey(),"");
                                            vertimo_zodziai.remove((Integer)mzv.getKey());

                                            surasta = 1;
                                            break;
                                    }
                            }
                    }
            }
            catch (IOException e)
                    {e.printStackTrace();}
    }
    //Vertimo varianto zodziai (tarp ju ir sinonimai)
    //buvo priskirti karkaso zodziams. Toliau, like zodziai
    //yra istatomi i likusias vietas arba karkasas yra prapleciamas.
        s = vertimas_copy.entrySet();
        Map<Integer,String>
        vertimas_map_surikiuotas;
            Iterator it1 = null;
            Map.Entry m = null;

            for (int k = 0; k < vertimas.size(); k++)
            if (vertimas.get(k).equals("")) {
                    vertimas.remove(k);
                    --k;
            }

            for (int i = 0; i < vertimas.size(); i++) {
                    int max_index = -1;
                    Set s1 = vertimas_map.entrySet();
                    it1 = s1.iterator();
                    vertimas_map_surikiuotas = new
           LinkedHashMap<Integer,String>();
                    while (it1.hasNext()) {
                                    Map.Entry m1 =(Map.Entry)it1.next();
                                    if ((Integer)m1.getKey() > max_index)
                                    max_index = (Integer)m1.getKey();
                    }

                    for (int k = 0; k <= max_index; k++)
                            if (vertimas_map.containsKey(k))
                            vertimas_map_surikiuotas.put(
                                    k,vertimas_map.get(k));

                    vertimas_map = vertimas_map_surikiuotas;

                    it = s.iterator();
            it1 = s1.iterator();
            int indeksas = 0,
                            indeksas_m = -1, indeksas_d = -1;
            int pozicija = 0;

            while (it.hasNext()) {
                    m =(Map.Entry)it.next();
                    if (vertimas.get(i).toLowerCase().equals(
                                    ((String)m.getValue()).toLowerCase()))
                            break;
                    indeksas++;
            }

            while (it1.hasNext()) {
                    m =(Map.Entry)it1.next();
```

```
		if (indeksas == 0) {
			if (
			vertimas_copy.get(indeksas+1).toLowerCase().equals(
				((String)m.getValue()).toLowerCase()))
			{indeksas = (Integer)m.getKey(); break;}
		}
		else if (indeksas+1 == vertimas_copy.size()) {
			if (vertimas_copy.get(indeksas-1).toLowerCase().equals(
			((String)m.getValue()).toLowerCase()))
			{indeksas = (Integer)m.getKey(); break;}
		}
		else {
			if (vertimas_copy.get(indeksas + 1).toLowerCase().equals(
			((String)m.getValue()).toLowerCase()))
			{indeksas_m = (Integer)m.getKey();}

			if (vertimas_copy.get(indeksas - 1).toLowerCase().equals(
			((String)m.getValue()).toLowerCase()))
			{indeksas_d = (Integer)m.getKey();}
		}
	}

	vertimas_map_new =
			new LinkedHashMap<Integer, String>();
	karkasas_map_new =
			new LinkedHashMap<Integer, String>();
	it1 = s1.iterator();
	Vector<Integer> laisvi = new Vector<Integer>();
int max_key = -1;
sumaisymo_tinklas.put(sistema,vertimas_map);

		while (it1.hasNext()) {
				Map.Entry m1 =(Map.Entry)it1.next();
				if ((Integer)m1.getKey() > max_key)
				max_key = (Integer)m1.getKey();
		}

		Set s2 = sumaisymo_tinklas.get(0).entrySet();
Iterator it2 = s2.iterator();

		if (vertimas_map.size()
				< sumaisymo_tinklas.get(0).size()) {
			while (it2.hasNext()) {
				Map.Entry m2 =(Map.Entry)it2.next();
				if (!vertimas_map.containsKey(
					(Integer)m2.getKey()))
				laisvi.add((Integer)m2.getKey());
			}
		}
		//Jeigu ne vienas zodis vertime nebuvo
		//priskirtas karkasui suteikiame jam karkaso dydi
		if (max_key == -1) {
			for (int k = 0; k < karkasas.size(); k++)
				laisvi.add(k);
		}

		int laisvu_poziciju_yra = 0;

		for (int k = 0; k < laisvi.size(); k++)	{
			if (laisvi.get(k) != null) {
				laisvu_poziciju_yra = 1; break;}
```

```
        }

if (indeksas_d != -1 &&
                !vertimas_map.containsKey(
                        indeksas+1) &&
                        !vertimas_map.containsKey(
                                (Integer)m.getKey()+1)) {
                vertimas_map.put(
                        (Integer)m.getKey()+1,
                        vertimas.get(i));
                }
else if (indeksas_m != -1 &&
                !vertimas_map.containsKey(
                        indeksas-1) &&
                        !vertimas_map.containsKey(
                                (Integer)m.getKey()-1)) {
                vertimas_map.put(
                        (Integer)m.getKey()-1,
                        vertimas.get(i));
                }
else if ((indeksas == 0 ||
                indeksas+1 == vertimas_copy.size() ||
                laisvi.size() == 0)
                && laisvu_poziciju_yra == 0) {
        if (indeksas+pozicija == -1)
                indeksas = 1;
        if (pozicija == 0)
                pozicija = 1;
        if (indeksas+1 == vertimas_copy.size())
                pozicija = 0;

int max_vertimo_ilgis = -1;
for (int var = 0; var < sumaisymo_tinklas.size(); var++) {
        if (sumaisymo_tinklas.get(var).size() > max_vertimo_ilgis)
                max_vertimo_ilgis =
                        sumaisymo_tinklas.get(var).size();
}

String temp = "", temp_k = "";
for (int sys = sumaisymo_tinklas.size()-1;
                        sys > 0; sys--) {
        for (int k = 0; k <= max_vertimo_ilgis; k++) {

                        if (k < indeksas+pozicija) {
                        if (sumaisymo_tinklas.get(sys).get(k) != null)
                        vertimas_map_new.put(k,
                                (String)sumaisymo_tinklas.get(sys).get(k));
                        if (sumaisymo_tinklas.get(0).get(k) != null)
                        karkasas_map_new.put(k,
                                (String)sumaisymo_tinklas.get(0).get(k));
                        }
                        else if (k == indeksas+pozicija) {
                                if (sys == sistema) {
                                        if (
                                        sumaisymo_tinklas.get(sys).containsKey(k)
                                        &&
                                        sumaisymo_tinklas.get(sys).get(k) != null)
                                        temp = (String)
                                        sumaisymo_tinklas.get(sys).get(k);
                                        vertimas_map_new.put
                                        (k,     vertimas.get(i));
                                        if (
```

```
                        sumaisymo_tinklas.get(0).containsKey(k)
                        &&
                        sumaisymo_tinklas.get(0).get(k) != null)
                        temp_k = (String)
                        sumaisymo_tinklas.get(0).get(k);
                        karkasas_map_new.put(k,"");
                }
                else if (!temp.isEmpty() ||
                                !temp_k.isEmpty()) {
                if (sumaisymo_tinklas.get(sys).get(k) != null)
                vertimas_map_new.put(k+1,
                (String)sumaisymo_tinklas.get(sys).get(k));
                if (sumaisymo_tinklas.get(0).get(k) != null)
                karkasas_map_new.put(k,
                (String)sumaisymo_tinklas.get(0).get(k));
                }
                else {
                if (sumaisymo_tinklas.get(sys).get(k) != null)
                vertimas_map_new.put(k,
                                (String)sumaisymo_tinklas.get(sys).get(k));
                if (sumaisymo_tinklas.get(0).get(k) != null)
                karkasas_map_new.put(k,
                                (String)sumaisymo_tinklas.get(0).get(k));
                }
        }
        else {
                if (!temp.isEmpty() ||
                                !temp_k.isEmpty()) {
                        if (sys == sistema) {
                        String temp_old = temp;
                        String temp_k_old = temp_k;
                        if
                        (sumaisymo_tinklas.get(sys).containsKey(k))
                        temp = (String)
                                        sumaisymo_tinklas.get(sys).get(k);
                        vertimas_map_new.put(k,temp_old);
                        if
                        (sumaisymo_tinklas.get(0).containsKey(k))
                        temp_k = (String)
                                        sumaisymo_tinklas.get(0).get(k);
                        karkasas_map_new.put(k,temp_k_old);
                        }
                        else {
                                if (sumaisymo_tinklas.get(sys).get(k) != null)
                                vertimas_map_new.put(k+1,
                                (String)sumaisymo_tinklas.get(sys).get(k));
                                if (sumaisymo_tinklas.get(0).get(k) != null)
                                karkasas_map_new.put(k,
                                (String)sumaisymo_tinklas.get(0).get(k));

                        }
                }
                else {
                if (sumaisymo_tinklas.get(sys).get(k) != null)
                vertimas_map_new.put(k,
                                (String)sumaisymo_tinklas.get(sys).get(k));
                if (sumaisymo_tinklas.get(0).get(k) != null)
                karkasas_map_new.put(k,
                                (String)sumaisymo_tinklas.get(0).get(k));
                        }
        }
}
```

```
                sumaisymo_tinklas.put(0,karkasas_map_new);
                sumaisymo_tinklas.put(sys,vertimas_map_new);
                if (sys == sistema) {
                vertimas_map = vertimas_map_new;
                }
                        vertimas_map_new = new LinkedHashMap<Integer, String>();
        karkasas_map_new = new LinkedHashMap<Integer, String>();
        }
                }

    else {
                int maziausias_skirtumas = Integer.MAX_VALUE;
        int pozicija_k = 0;
        for (int k = 0; k < laisvi.size(); k++) {
                        if (Math.abs(
                                        laisvi.get(k) - indeksas)
                                        < maziausias_skirtumas) {
                        maziausias_skirtumas = Math.abs(
                                        laisvi.get(k) - indeksas);
                        pozicija_k = laisvi.get(k);
                        }
                }
        vertimas_map.put(
                                pozicija_k,
                                                vertimas.get(i));
        }
        sumaisymo_tinklas.put(sistema,vertimas_map);
        }
        return vertimas_map;
        }
}
```