TARTU ÜLIKOOL
MATEMAATIKA-INFORMAATIKATEADUSKOND
Arvutiteaduse instituut
Informaatika eriala

Ando Paju

UTF-8 kodeeringu toe lisamine programmile *Lingua::Ident*

Bakalaureusetöö (6 EAP)

Juhendaja: prof. Heiki-Jaan Kaalep

Autor: . . . . . " . . . . . " august 2013
Juhendaja: . . . . . " . . . . . " august 2013
Lubada kaitsmisele
Professor: . . . . . " . . . . . " august 2013

Tartu 2013

Sisukord

Sissejuhatus....3
1. Meetodid keele tuvastamiseks....5
2. Markovi mudel ja Ted Dunningu algoritm....6
    2.1 Markovi mudel....6
    2.2 Ted Dunningu algoritm....7
3. Lingua::Ident....8
    3.1 Lingua::Ident tutvustus....8
    3.2 Puudujäägid praeguses programmis....8
    3.3 Pakutud lahendus....9
4. Katse....10
    4.1 Lähteandmed....10
    4.2 Katse läbiviimine....10
    4.3 Kokkuvõte katsetustulemustest....11
Kokkuvõte....13
Summary....14
Kasutatud kirjandus....15
Lisad....17
    Lisa 1. DVD bakalaureusetöö juurde kuuluvate failidega....17
    Lisa 2. Erinevate keeletuvastusmeetodite võrdlus....19
    Lisa 3. Õigesti klassifitseeritud laused....20
    Lisa 4. Valesti klassifitseeritud laused....21
    Lisa 5. Kõrvale jäetud laused....22

Sissejuhatus

Suur osa internetist on ingliskeelne, samas väheneb ingliskeelse teksti osatähtsus aasta-aastalt. Kui arvestada, et keel võib leheküljel olla valesti märgitud, siis on teksti tegeliku keele leidmine raskem. Keelt on vaja tuvastada mitmel puhul, näiteks enne kui teksti on võimalik tõlkida, tuleb selle keel kõigepealt tuvastada. Teine näide on teksti automaatne kategoriseerimine vastavalt keelele. Mõlemal juhul on abiks keeletuvastus.

Inimesel on kerge öelda, kas üks sõna on konkreetses keeles või mitte. Seejuures ei pea inimene teadma sõna enda tähendust, piisab vaid sellest, et sõna oleks teistele selle keele sõnadele piisavalt sarnane. Seega saab inimene, kes on mõnda keelt vaid veidi kohanud vaevata öelda, kas sõna või lause võiks sinna keelde kuuluda.

Arvuti jaoks on keele tuvastamine natuke raskem. Et saavutada piisavalt häid tulemusi ka lühikesel tekstil, tuleb programmi treenida suurel lausetehulgal. Põhiline raskus tekib sellest, kui treeninglausete seas on teisest keelest sõna või võõrsõna. Inimese jaoks on see väiksem probleem, kuna inimese mõtlemine sobib mustrite (s.o. üksikute mittesobivate sõnade või lausete) leidmiseks paremini. Arvuti jaoks on probleemiks ka tuvastamine, et sõna pole etteantud treeningkeelte hulgas. Sel juhul kipub programm eelistama mõnda treeningkeelt.

Eelnevalt on olemas vabavaraline programm *Lingua::Ident* [1] keeletuvastuseks, mis töötab baitidel. UTF-8 kodeeringus on aga mitmebaidilisi sümboleid [2], mis suurendavad müra ja halvendavad keeletuvastust. Käesoleva bakalaureusetöö eesmärk on teada saada, kas UTF-8 sümbolites peituva lisainfo arvestamine parandab eesti keele tuvastamist.

Töö esimeses peatükis võrdlen erinevaid meetodeid keele tuvastamiseks ja valin neist ühe, millega selles töös tegelema hakkan.

Teises peatükis selgitan valitud meetodi tööpõhimõtet.

Töö kolmandas peatükis tutvustan programmi, mis valitud meetodit rakendab — *Lingua::Ident.*

Neljandas peatükis täiendan programmi *Lingua::Ident*, selliselt, et ta toetaks UTF-8 kodeeringus sümboleid ja viin läbi katse, et näha kas muudetud programm suudab eestikeelseid sõnu paremini tuvastada, kuna sel juhul ei jää UTF-8 sümbolid treeningul kõrvale, vaid nendest on programmi treenimisel kasu.

Töö lisana on kaasas DVD, millel on töö käigus muudetud *Lingua::Ident* failid, tuvastamiseks kasutatud ning tuvastatavad tekstid. Lisaks on toodud välja katsetustulemused.

1. Meetodid keele tuvastamiseks

Keeletuvastuseks on loodud mitmeid meetodeid. Üksteisest erinevad nad põhiliselt paindlikkuse ja keerukuse poolest. Näiteks tundub tuvastamiseks hea valik sõnastik, kus on kõik tuvastatava keele sõnad sees või selline, kus on ainult sellele keelele ainuomased sõnad. Mingil määral selline lähenemine toimibki, kuid ei suuda toime tulla kirjavigade või lühemate tekstidega, lisaks on probleeme sõnavormide ja uute sõnadega. Sellistel põhjustel ei ole sõnastiku kasutamine üldiselt piisavalt paindlik.

Parem on kasutada keelele iseäralikke statistilisi omadusi. Keele tuvastamiseks on sel juhul vaja võrrelda keele omadusi teksti omadustega ning sobivaim valida. Lihtsaim neist on tähtede esinemissagedusel põhinev meetod. Nimelt kasutavad keeled tähti erineva sagedusega, mis võimaldab neid üksteisest eristada. See meetod kannatab aga sama vea all ehk vajab pikka teksti. Lühema teksti jaoks sobivad keerukamad meetodid, näiteks Markovi mudelit kasutav algoritm (Dunning) [3] või N-grammidel põhinev meetod (Cavnar) [4]. Põhiliste keeletuvastusmeetodite võrdluseks vt. Lisa 2. Tabel 1.

Valisin keeletuvastuseks Ted Dunningu välja töötatud algoritmi, kuna see suudab juba väheste treeningandmetega tuvastada suure keele suure kindlusega. Lisaks on programmeerimiskeeles Perl olemas seda rakendav programm.

2. Markovi mudel ja Ted Dunningu algoritm

2.1 Markovi mudel

Markovi mudel on matemaatiline mudel, mis leiab tuleviku sündmuse toimumise tõenäosuse arvestades eelnenud mustreid. Mudel jagab modelleeritava erinevateks olekuteks ja leiab tõenäosused ühest olekust teise minekuks.

Markovi mudeli abil saab modelleerida vaid selliseid sündmusi, mis sõltuvad ainult eelnevatest olekutest. Keeletuvastuse puhul on olekuks täht või mitu tähte. Eelnevad tähed mõjutavad neile järgnevate tähtede esinemist keeles (keele tähed pole juhuslikus järjekorras vaid esinevad teatud viisil korrapäraselt). Näiteks on tekstilõigu *karu karjus* kõik esimese astme Markovi mudeli olekud *k*, *a*, *r*, *u*, *j* ja *s*.

Esimese astme Markovi mudel on selline, kus järgmise oleku tõenäosuse määramiseks piisab ainult eelmise oleku arvestamisest.

Valemina näeb see välja nii:

$$P(c_k|c_0, c_1, \ldots, c_{k-1}) = P(c_k|c_{k-1})$$

kus c_k on *k*-s olek või keeletuvastuse puhul *k*-s täht.

Üldiselt on *n*-astme Markovi mudel selline, mis arvestab *n* eelnevat olekut, et määrata järgmine tähe tõenäosus:

$$P(c_k|c_0, c_1, \ldots, c_{k-1}) = P(c_k|c_{k-n}, \ldots, c_{k-1})$$

kusjuures *0*-astme mudel ei arvesta eelmisi olekuid, vaid ainult eelnevat tõenäosust (keeletuvastuse puhul vastab *0*-astme mudelile tähtede sagedus keeles) [5].

2.2 Ted Dunningu algoritm

Treeningtekstist loetakse tähemärk haaval k tähemärki korraga (k - astme Markovi mudeli puhul) ja märgitakse k tähemärgiliste sõnede esinemise sagedused. Kui terve dokument on niiviisi läbitud leitakse iga Markovi mudeli tõenäosusteks järgmise valemi abil:

$$p(w_1 \dots w_{k+1}) = \frac{T(w_1 \dots w_{k+1}) + 1}{T(w_1 \dots w_k) + |A|} \quad ,$$

kus $|A|$ on tähestiku suurus, $T(w_1 \dots w_k)$ on k-tähemärgilise sõne esinemiste koguarv, $T(w_1 \dots w_{k+1})$ vastavalt terve $k+1$ pikkusega sõne esinemiste koguarv ja $p(w_1 \dots w_{k+1})$ on $k+1$tähemärgilise sõne esinemise tõenäosus.

Kui on vaja teksti tuvastada, siis kasutatakse teksti kategoriseerimiseks järgmist hinnangufunktsiooni:

$$\log p = \sum_{w_1 \dots w_{k+1} \in S} T(w_1 \dots w_{k+1}) \log p(w_{k+1}, w_1 \dots w_k)$$

kus $T(w_1 \dots w_{k+1})$ on Markovi mudeli olekute koguarv ja $p(w_{k+1} | w_1 \dots w_k)$ on tõenäosus iga konkreetse oleku kohta.

Iga treenitud keele kohta leitakse sellele vastav arvuline väärtus ja keel mis on lähim nullile tagastatakse kui kõige tõenäolisem keel.

Sellise kategoriseerimise tulemusena saab hinnata, kui suure tõenäosusega kuulub tekst mõnda treenitavatest keeltest. Kui teksti algkeel pole treenitud keelte hulgas, siis algoritm selle keele tõenäosuse kohta midagi arvata ei suuda [3].

3. Lingua::Ident

3.1 Lingua::Ident tutvustus

Lingua::Ident on programmeerimiskeeles Perl kirjutatud vabavaraline teek [1], mis tuvastab sõna, kasutades Markovi mudelit. Programm on Michael Piotrowski poolt kirjutatud ja kasutab Ted Dunningu 1994 välja töötatud algoritmi veidi muudetud varianti. Põhiline erinevus on selles, et *Lingua::Ident* leiab ainult bi- ja trigrammide (bigramm on kahe-, trigramm kolmetäheline sõne) tõenäosused, kuna pikemate sõnede kasutamine lühikese treeningteksti puhul tulemust ei paranda, vaid võivad põhjustada ületreeningut [3].

Programmi kasutamiseks on vaja bi- ja trigrammide üleminekumaatrikseid, mille tegemiseks sobib kaasas olev *trainlid* utiliit (teeb üleminekumaatriksid sisendist). Kui need on olemas võtab *Lingua::Ident* sisendiks testitava tekstilõigu ja annab väljundiks tõenäosused iga keele kohta, mis talle *trainlid* programmi abil on selgeks õpetatud.

Programm töötab inglisekeelsel tekstil hästi, tuvastades 20-baidist teksti õigesti 92 % täpsusega kasutades 50kB treeningandmeteid ning 99,9 % täpsusega 500-baidise teksti tuvastamisel [3].

3.2 Puudujäägid praeguses programmis

UTF-8 kodeering sisaldab tähemärke, mida *Lingua::Ident* arvestada ei oska. Nimelt töötab *Lingua::Ident* ühebaidiste tähemärkidega, loeb sisendit ning tekitab treeningfaile ASCII kodeeringus. Kui tuvastatav tekst on UTF-8 kodeeringus, siis võib see vähendada keeletuvastuse efektiivsust, kuna ühte märki loetakse mitmeks.

Kuna tõenäosusi arvutatakse tri- ja bigramme kasutades, siis on treeningu tulemusel failis palju ebatäpseid ridu. Näiteks saab UTF-8 kodeeringus „*õ,ä,ö,ü*“-st ASCII kodeeringus vastavalt „*Ãµ, Ã¤, Ã¶, Ã¼*“ (saadud kasutades [6]).

Kuivõrd ülejäänud tähed, peale ülalmainitud eesti keele tähtede ning *š* ja *ž*, on UTF-8 kodeeringus samad mis ASCII kodeeringus, siis toimib programm ka eesti keele puhul.

Sellegi poolest, kuna programm ei tea, et „*Ãµ*“ peaks olema üks täht („*õ*“), siis arvestab

ta selle kaheks. Kui need programmi jaoks kaks tähte satuvad kahe tri- või bigrammi algusse või lõppu, siis poolitab programm nad kahe tri- või bigrammi vahel ära. See aga muudab bi- ja trigrammide tõenäosusi ning suurendab seeläbi ebatäpsust keele tuvastamisel.

3.3 Pakutud lahendus

Pakutud lahendus on programmi muutmine nii, et see kasutaks UTF-8 kodeeringut nii treeningandmete väljundiks oleva faili kirjutamiseks kui ka keele hinnangut arvutades. Selleks tuleb muuta *Lingua::Ident* treeningmoodulit *trainlid* ning peafaili *Ident.pm*.

4. Katse

4.1 Lähteandmed

Katse eesmärgiks on teada saada, kuidas tuvastab muudetud programm eesti keelt võrreldes algse programmiga.

Treenisin programmi eesti, inglise ja saksa keelel. Programmi treenisin inglise ja saksa keelel, et oleks võimalik eristada mitte-eestikeelne tekst eestikeelsest. Eesti keele treenimiseks kasutasin Eesti Ekspressi 1996 aasta artiklite kogu [11]. Inglise keele treenimiseks kasutasin Brennan-Greenstadti korpust ja *Lingua::Ident'*iga kaasa tulnud näidisfaili *sample.en*; saksa keele treenimiseks *Lingua::Ident'*iga kaasa tulnud näidisfaili *sample.de.utf-8* ning lisaks ajakirja Frankfurter Rundshau põhjal tehtud korpust [7].

Tuvastatavad failid on üheksa delfi artiklite kommentaarid (*delfi1.xml-delfi9.xml*) XML [8] kujul. Tuvastatavad failid on UTF-8 kodeeringus ja enamjaolt eesti keeles, aga sisaldavad ka muid keeli.

Kõik failid on lisatud tööga kaasas olevale *DVD'*le.

4.2 Katse läbiviimine

Kõigepealt kasutasin programmiga *Lingua::Ident* kaasasolevat *trainlid* utiliiti, et treeningfailidest moodustada bi- ja trigramme, eelnevalt eemaldasin ebavajalikud märgid. Ebavajalikud on näiteks suurtähed, mille teisendasin väiketähtedeks. Lisaks eemaldasin üleliigsed tühikud, muud kirjavahemärgid ja sümbolid. Alles jäid ainult *Unicode* märgid, millel on omadus *Alphabetic* [9].

Seejärel lugesin failid, mille keelt on vaja tuvastada sisse, eemaldasin sobimatud märgid ja kasutasin *Lingua::Ident*'i meetodit *calculate()*, mis annab iga keele jaoks hinnangu, et tuvastatav fail on selles keeles.

Seejärel muutsin *trainlid* utiliiti nii, et see väljastaks UTF-8 kodeeringus

üleminekumaatriksi.

Iga keele jaoks saadud treeningfailid salvestasin kahte faili, üks muudetud ja teine algset programmi kasutades saadud fail (vastavalt „eesti_utf.trainlid“, „inglise_utf.trainlid“, „saksa_utf.trainlid“ ja „eesti.trainlid“, „inglise.trainlid“, „saksa.trainlid“).

Lisasin *Lingua::Ident*'ile muudetud kodeeringus treeningfailide toe ja muutsin programmi nii, et hinnangut arvutataks tähemärkidel mitte baidikaupa. Seejärel eemaldasin ebavajalikud märgid ja kordasin keeletuvastust samal viisil nagu algse programmiga.

Kuna tuvastavate lausete hulk oli suur ja eesmärk oli hinnata kuidas programm töötab pärast muudatusi, siis analüüsisin muudetud programmmi pakutud keeli vaid neile lausetele, milles algne ja muudetud programm on eri meelel. Kasutasin *Linux*'i utiliiti *diff* [10] ja salvestasin erinevused (DVD-l olemas failis *erinevused.diff*).

4.3 Kokkuvõte katsetustulemustest

Kuna sama sõne tuvastab programm alati samamoodi, arvestasin iga unikaalset sõne vaid üks kord, et korduvate lausete tõttu katse tulemus mõjutatud ei oleks.

Kokku oli üheksa delfi kommentaarifaili peale tuvastatavaid lauseid 29 055.

Leidus mitmeid lauseid, mida ei saanud klassifitseerida, kuna nad pole õiged sõnad või nad pole arusaadavad. Näiteks „böö“, „pähh“, „õõh“, „lõrr“, „ik õpa“ jne. Esimesi neist võiks veel eestikeelseks pidada, kuid leidsin et need on mõistlikum lihtsustamiseks kõrvale jätta. Lisaks ei saanud klassifitseerida lauseid, mis sisaldavad mitut keelt. Näiteks „to harry rasvtõbine“, „eefraim esimene“ jne. Kokku oli kümne delfi faili peale selliseid lauseid 13. Sellised laused jätsin analüüsi lihtsustamiseks samuti kõrvale (vt. Lisa 5.).

Eestikeelseid lauseid, mida muudetud programm erinevalt algsest programmist eestikeelseks pidas oli 18.

Mitte eestikeelseid lauseid tuvastas programm õigesti mitte eestikeelseteks kuuel korral.

Kordi, kus tekst oli eesti keeles, aga muudetud programm tuvastas mitte-eestikeelsena oli kuus.

Programm ei määranud testlausete hulgal ühelgi korral eesti keelt mitte-eestikeelsele

lausele.

Jättes kõrvale laused mida ei saanud kergesti klassifitseerida, tuvastas muudetud programm 24 võrra lauseid paremini (eesti keel tuvastatud eestikeelseks ja mitte-eesti mitte-eestikeelseks, vt. Lisa 3. Tabel 2) ja 6 võrra lauseid halvemini(mitte-eestikeelne lause eesti keelseks vt. Lisa 4. Tabel 3). Kokku tuvastas muudetud programm lauseid 18-e võrra paremini kui algne.

Sellest võib järeldada, et UTF-8 kodeeringu kasutamine parandab eesti keele tuvastamist vähesel määral. Kuna UTF-8 kodeeringu kasutamine baithaaval lugemise asemel suurendab programmi keerukust, aga märkimisväärset tulemust ei anna, siis pole see üldiselt otstarbekas viis keeletuvastust parandada.

Lisaks väärib mainimist, et enamik lauseid, kus muudetud programm algsest erines sisaldasid mitmebaidilisi täpitähti ja olid võrdlemisi lühikesed. Sellest võib järeldada, et programmist oleks kasu keelele, kus on suuremal hulgal mitmebaidilisi UTF-8 sümboleid ja sedagi lühemate tekstide puhul, pikema teksti korral piisab vaid baitide arvestamisest.

Kokkuvõte

Töö eesmärgiks oli leida, kas UTF-8 sümbolites peituva lisainfo arvestamine programmis *Lingua::Ident* parandab eesti keele tuvastamist. Hetkel kasutab *Lingua::Ident* keeltele hinnangu andmiseks baite.

Töö esimeses peatükis võrdlesin erinevaid keeletuvastuse meetodeid ja valisin Ted Dunningu algoritmi, mis kasutab Markovi mudelit.

Töö teises peatükis selgitasin, mida kujutab endast Markovi mudel ja Ted Dunningu algoritm.

Kolmandas peatükis leidsin, mis on *Lingua::Ident*'i puudused eesti keele jaoks ja pakkusin muudatused, mida sisse viia, et täpitähti (ja muid sümboleid, mida algses ASCII kodeeringus pole) ja UTF-8 kodeeringut arvestada oskaks.

Neljandas peatükis viisin muudatused programmi sisse ning korraldasin katse, et näha kas muudetud programm tuvastab esialgsest programmist eesti keelt paremini.

Katse tulemusena leidsin, et UTF-8 kasutamine baitide asemel aitas programmil veidi paremini eesti keelt tuvastada. Keeletuvastusel on tõenäoliselt rohkem kasu selliste keelte jaoks, mis kasutavad suuremal hulgal mitmebaidilisi UTF-8 sümboleid.

Adding UTF-8 Encoding Support to the Program Lingua::Ident

Bachelor Thesis

6 ECTS

Ando Paju

Summary

The purpose of this paper is to analyze whether adding UTF-8 encoding support to *Lingua::Ident* will provide any benefits. Currently *Lingua::Ident* uses bytes internally to decide how each language is rated.

In the first paragraph I gave overview of current language identification methods and chose the algorithm developed by Ted Dunning which uses Markov models as the basis for this paper.

In the second paragraph I explained what is a Markov model and how does Dunning's algorithm work.

In the third paragraph possible disadvantages of Lingua::Ident for the Estonian language were listed and proposed what changes should be implemented to use umlauts (and other characters not present in the original ASCII encoding) for language identification in UTF-8 encoded documents.

Fourth paragraph contains experiments with the changed *Lingua::Ident*, to see whether adding encoding support made any difference.

Experiment results concluded that adding UTF-8 encoding support to *Lingua::Ident* provided minor benefit to identify the Estonian language. Benefits of language identification are probably greater for languages that use more multi-byte UTF-8 symbols.

Kasutatud kirjandus

Kõik allikad on kontrollitud 14.08.2013

1. M. Piotrowski, *Lingua::Ident*

   http://search.cpan.org/~mpiotr/Lingua-Ident-1.7/Ident.pm

2. *UTF-8*

   http://en.wikipedia.org/wiki/UTF-8

3. T. Dunning, *Statistical Identification of language*, Computing Research Laboratory, New Mexico State University, 1994

4. W.B Cavnar, J.M. Trenkle, *N-gram-based text categorization,* Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, USA 1994

5. P. Vojtek, M. Bielikov´a, *Comparing Natural Language Identification Methods based on Markov Processes*, Slovakkia, 2007

6. *Character set converter*

   http://kanjidict.stc.cx/recode.php

7. *Frankfurter Rundshau artiklite kogu*

   http://www.elsnet.org/resources/eci-sample-german.txt

8. *Extensible Markup Language (XML) 1.0 (Fifth Edition)*

   http://www.w3.org/TR/2008/REC-xml-20081126

9. *Unicode Character Database*

   http://www.unicode.org/Public/5.1.0/ucd/UCD.html#Alphabetic

10. *Commands & Utilities Reference, The Single UNIX® Specification, Issue 7,* The Open Group

    http://pubs.opengroup.org/onlinepubs/9699919799/utilities/diff.html

11. *Eesti Ekspressi failid XML-kujul*, arhiivist *Ekspress.zip*, fail *aja_ee_1996_33.xml* http://www.cl.ut.ee/korpused/segakorpus/ekspress/index.php?lang=et

Lisad

Lisa 1. DVD bakalaureusetöö juurde kuuluvate failidega

Bakalaureusetöö juurde kuuluv DVD sisaldab järgmiseid faile:

tuvasta_keel.pl — *Lingua::Ident*'i kasutav programm

Ident_muudetud.pm — muudetud *Lingua::Ident*'i fail, millele on lisatud UTF-8 tugi

trainlid.pl — muudetud *Lingua::Ident*'i fail, millele on lisatud UTF-8 tugi

eesti.trainlid — eesti keele treenimisel saadud fail

inglise.trainlid — inglise keele treenimisel saadud fail

saksa.trainlid — saksa keele treenimisel saadud fail

eesti_utf.trainlid — eesti keele treenimisel saadud fail UTF-8 kodeeringus

inglise_utf.trainlid — eesti keele treenimisel saadud fail UTF-8 kodeeringus

saksa_utf.trainlid — eesti keele treenimisel saadud fail UTF-8 kodeeringus

aja_ee_1996_33.lause-real — Eesti Ekspressi 1996 aasta artiklite kogu

sample.en — *Lingua::Ident*'iga kaasa tulnud näidisfail

brown_ajakirjandus.txt — Brennan-Greenstadti korpus

sample.de.utf-8 — *Lingua::Ident*'iga kaasa tulnud näidisfail

eci-sample-german.txt — ajakirja Frankfurter Rundshau artiklite kogu

kommentaarid/

- delfi1.xml — delfi kommentaarid XML kujul
- delfi2.xml — delfi kommentaarid XML kujul
- delfi3.xml — delfi kommentaarid XML kujul
- delfi4.xml — delfi kommentaarid XML kujul
- delfi5.xml — delfi kommentaarid XML kujul

delfi6.xml — delfi kommentaarid XML kujul

delfi7.xml — delfi kommentaarid XML kujul

delfi8.xml — delfi kommentaarid XML kujul

delfi9.xml — delfi kommentaarid XML kujul

kasutusjuhend.pdf — kirjutatud programmide kasutusjuhendid

erinevused.diff — leitud erinevused muudetud programmi ja algse programmi vahel

tuvastafailid.sh — tuvastab alamkaustades olevate delfi kommentaaride keele (eesti või mitte-eesti)

Lisa 2. Erinevate keeletuvastusmeetodite võrdlus

Meetod	Eelised	Puudused
Sõnastikku kasutavad meetodid (lühimad, unikaalsed, sagedaimad sõnad jne.)	• kerge rakendada • toimib hästi pika tekstilõigu puhul	• vajab pikka treeningteksti • ei tuvasta uusi sõnu ega sõnavorme • ei suuda toime tulla kirjavigadega • pole usaldatav kui tekst on lühike
Markovi mudelit kasutav algoritm (Dunning)	• on usaldusväärne ka lühikese tekstiga (20 tähemärki või rohkem)	• keerulisem rakendada
N-grammidel põhinev meetod (Cavnar)	• on usaldusväärne ka lühikese tekstiga (20 tähemärki või rohkem)	• keerulisem rakendada
Tähtede esinemissagedustel põhinev meetod	• väike maht iga keele kohta • kerge rakendada	• vajab pikemat teksti, et keeli eristada • väike usaldusväärsus
Naiivne Bayesi klassifitseerija, tugivektor-masinad (SVM — *support vector machines*) või k-lähima naabri meetod (kNN — k-Nearest Neigbour) [5]	• suhteliselt väike maht iga keele kohta	• tuvastatav tekst vajab eeltöötlust, näiteks sõnade eemaldamine (sidesõnad vms.) • keerulisem rakendada

Tabel 1. Erinevate keeletuvastusmeetodite võrdlus

Lisa 3. Õigesti klassifitseeritud laused

Lause	Lause keel	Muudetud programmiga tuvastatud keel
soku is back jag fick min öl	mitte-eesti	mitte-eesti
du är en hurri	mitte-eesti	mitte-eesti
kära nabolandet	mitte-eesti	mitte-eesti
elagu relvamüüjad	eesti	eesti
jobu	eesti	eesti
ahjaa hr kingo	eesti	eesti
xaetärrud	mitte-eesti	mitte-eesti
ämbrite vürst	eesti	eesti
andestame	eesti	eesti
aitäh kebab	eesti	eesti
vana hästi vana m u i n a s j u t t	eesti	eesti
aitähh	eesti	eesti
muudkui jõuram jõuram jõuram	eesti	eesti
õhhhh lasteaed	eesti	eesti
tolku malo	mitte-eesti	mitte-eesti
ooi jõuram nii leebe täna	eesti	eesti
viinaaaa viinaaa muidu nõrkkeen maa	eesti	eesti
hääduutjah	eesti	eesti
eimaeitea	eesti	eesti
või nummerdada kylli kylli kylli	eesti	eesti
baarikärbes aitüma	eesti	mitte-eesti
veebruarmärts	eesti	eesti
märts	eesti	eesti
või sina nõuad õigust	eesti	eesti

Tabel 2. Õigesti klassifitseeritud laused

Lisa 4. Valesti klassifitseeritud laused

Lause	Lause keel	Muudetud programmiga tuvastatud keel
kõver	eesti	mitte-eesti
vait te könnid	eesti	mitte-eesti
kõtt nurka	eesti	mitte-eesti
müstika	eesti	mitte-eesti
õige jah	eesti	mitte-eesti
inimnääpsud	eesti	mitte-eesti

Tabel 3. Valesti klassifitseeritud laused

Lisa 5. Kõrvale jäetud laused

eefraim esimene

lõrr

põmmm

ttüür

ööörtf

ik õpa

ööäkk

bööööö

bööööö zzzz

bööööööööööööööööööööööö

pähh

to harry rasvtõbine

ardrew floyd webber soku läheb ölle järgi

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, Ando Paju

(sünnikuupäev: 23.12.1989)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose
   UTF-8 kodeeringu toe lisamine programmile Lingua::Ident,

mille juhendaja on Heiki-Jaan Kaalep,

1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **14.08.2013**