Kauno technologijos universitetas

Informatikos fakultetas

Programavimo uždavinių sudėtingumo automatinio įvertinimo tyrimas

Baigiamasis magistro studijų projektas

Artūras Skarbalius

Projekto autorius

doc. Mantas Lukoševičius

Vadovas

Kaunas, 2021

Kauno technologijos universitetas

Informatikos fakultetas

Programavimo uždavinių sudėtingumo automatinio įvertinimo tyrimas

Baigiamasis magistro studijų projektas

Programų sistemų inžinerija (6211BX011)

Artūras Skarbalius

Projekto autorius

doc. Mantas Lukoševičius

Vadovas

doc. Šarūnas Packevičius

Recenzentas

Kaunas, 2021

Kauno technologijos universitetas

Informatikos fakultetas

Artūras Skarbalius

Programavimo uždavinių sudėtingumo automatinio įvertinimo tyrimas

Akademinio sąžiningumo deklaracija

Patvirtinu, kad:

1. baigiamąjį projektą parengiau savarankiškai ir sąžiningai, nepažeisdama(s) kitų asmenų autoriaus ar kitų teisių, laikydamasi(s) Lietuvos Respublikos autorių teisių ir gretutinių teisių įstatymo nuostatų, Kauno technologijos universiteto (toliau – Universitetas) intelektinės nuosavybės valdymo ir perdavimo nuostatų bei Universiteto akademinės etikos kodekse nustatytų etikos reikalavimų;

2. baigiamajame projekte visi pateikti duomenys ir tyrimų rezultatai yra teisingi ir gauti teisėtai, nei viena šio projekto dalis nėra plagijuota nuo jokių spausdintinių ar elektroninių šaltinių, visos baigiamojo projekto tekste pateiktos citatos ir nuorodos yra nurodytos literatūros sąraše;

3. įstatymų nenumatytų piniginių sumų už baigiamąjį projektą ar jo dalis niekam nesu mokėjęs (-usi);

4. suprantu, kad išaiškėjus nesąžiningumo ar kitų asmenų teisių pažeidimo faktui, man bus taikomos akademinės nuobaudos pagal Universitete galiojančią tvarką ir būsiu pašalinta(s) iš Universiteto, o baigiamasis projektas gali būti pateiktas Akademinės etikos ir procedūrų kontrolieriaus tarnybai nagrinėjant galimą akademinės etikos pažeidimą.

Artūras Skarbalius

Patvirtinta elektroniniu būdu

Kauno technologijos universitetas

Informatikos fakultetas

Baigiamojo magistro projekto užduotis

Projekto tema	Programavimo uždavinių sudėtingumo automatinio įvertinimo tyrimas
Reikalavimai ir sąlygos (tikslinti pavadinimą pagal poreikį)	Kuriami mašininio mokymo modeliai, kurie skirti atrasti programavimo uždavinio sudėtingumą pagal vieną iš penkių kategorijų (pradinis, lengvas, vidutiniškas, sunkus, ekspertams).
Vadovas / Vadovė	(vadovo pareigos, vardas, pavardė, parašas) (data)

Skarbalius, Artūras. Programavimo uždavinių sudėtingumo automatinio įvertinimo tyrimas. Magistro studijų baigiamasis projektas / vadovas doc. Mantas Lukoševičius; Kauno technologijos universitetas, informatikos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Programų sistemų inžinerija .

Reikšminiai žodžiai: Programavimo problema, sudėtingumo įvertinimas, natūralios kalbos tvarkymas, konvoliucinis neuroninis tinklas, gilusis mokymas, mašininis mokymas.

Kaunas, 2021. 48 p.

Santrauka

Šiame dokumente bus aprašomas tyrimas, kurio metu buvo bandoma panaudoti mašininį mokymą automatiškai įvertinti programavimo uždavinių sudėtingumą. Įprastai šie uždaviniai turi tekstinį aprašymą ir kartu pridėtus paveikslėlius bei lenteles, taigi reikia įvertinti galimus modelius atskirai skirtingiems variantams. Projekto tikslas – išsiaiškinti, ar tokio tipo mašininį mokymą įmanoma naudoti programavimo uždavinių sunkumo įvertinimui.

Atlikto darbo metu buvo surinktas tinkamas duomenų rinkinys, ištiriami gaunami rezultatai pagal tekstą ir pagal paveikslėlius, tada bandoma sukurti modelį, kuris leis įvertinti abu vienu metu. Taip pat pabaigoje bus pridedami gauti rezultatai bei teoriniai būdai, kuriais būtų galima bandyti patobulinti sistemą.

Skarbalius, Artūras. Research on Automatic Difficulty Estimation of Programming Problems. Master's Final Degree Project / supervisor doc. Mantas Lukoševičius; Faculty of Informatics, Kaunas University of Technology.

Study field and area (study field group): Software Engineering.

Keywords: Programming problem, difficulty evaluation, natural language processing, convolutional neural network, deep learning, machine learning.

Kaunas, 2021. 48 pages.

Summary

This document details an attempt to use machine learning for automatic evaluation of the difficulty of programming problems or exercises. Typically, the problems consist of both text description and accompanying figures. As such, evaluation of various possible models for both text and images is required. The goal of this project is to determine whether this sort of machine learning can reliably evaluate the difficulty of programming exercises.

The work involves collecting and creating a suitable dataset, analyzing the evaluation based on the text and the image data separately, and a method of combining the two methods is then attempted. The results of this investigation are reported near the end of the document, alongside with potential theoretical methods to improve the system further.

Turinys

Lentelių sąrašas

Paveikslų sąrašas

Santrumpų ir terminų sąrašas

Santrumpos:

L-BFGS – Limited-memory Broyden–Fletcher–Goldfarb–Shanno;

Terminai:

Mašininis mokymas – pritaikymas bei studijavimas metodų, kurie „išmoksta“ iš didelių kiekių duomenų.

Natūralios kalbos tvarkymas – natūralių kalbų išmokymas ir pritaikymas kompiuteriams (pvz. sakinių supratimui, rašymui, t.t.).

Žinių vertinimas – nuolatinis informacijos apie pasiekimus kaupimo ir interpretavimo procesas.

Žinių įvertinimas – vertinimo proceso rezultatas; sprendimas apie mokinio pasiekimus.

Žinių įsivertinimas – paties asmens daromi sprendimai apie pažangą bei pasiekimus.

Įvadas

Šis dokumentas sukurtas pateikti sukurtą projektą, kuriame bus kuriama dirbtinį intelektą naudojančią užduočių sudėtingumo įvertinimo sistema. Dokumente pateikiamos kelios dalys:

- Teorinė dalis – aprašymas apie įvairias dalis, kurios svarbios šio tipo projektui:
    - Žinių vertinimo metodika
    - Buvę bandymai šią problemą išspręsti
    - Algoritmai tekstui
    - Algoritmai paveikslėliams
    - Galimi metodai apjungti modelius
- Projektinė dalis – informacija apie sukurtą projektą:
    - Reikalavimai ir architektūra
    - Informacija apie sukurtą tekstu paremtą sudėtingumo įvertinimą
    - Informacija apie sukurtą paveikslėliais paremtą sudėtingumo įvertinimą
    - Apjungtų modelių sudėtingumo įvertinimą
    - Informacija apie vartotojo sąsają
- Tyrimo dalis – aprašymas apie pakeitimus, kurie buvo bandyti projektui:
    - Teksto išankstinio apdorojimo pakeitimai
    - Paveikslėlių klasifikacijos modelių išbandymai
    - Dvišakių modelių testavimas

Įvairaus pobūdžio programavimo uždaviniai naudojami, siekiant išmokyti susidomėjusius programavimu kaip tai atlikti praktiniu būdu. Taip pat kai kuriais atvejais jie yra naudojami įvertinti arba demonstruoti asmens įgūdžius tam tikroje programavimo srityje.

Tačiau yra keletas šio tipo uždavinių problemų. Viena iš labiau pastebimų – programavimo uždaviniai labai priklauso nuo to, kas jį parašė, bei to, kas bando išspręsti. Dažnai nebūna objektyvaus būdo teisingai įvertinti uždavinio sudėtingumo, ypač dėl to, nes rašantysis turi žinoti teisingą atsakymą – neaišku kaip įvertinti, kai asmeniškai rašančiam uždavinį nekyla nesklandumų jį sprendžiant.

Šis projektas bando išspręsti dalį šios problemos – naudojantis mašininiu mokymu, bandoma įvertinti programavimo uždavinių sudėtingumą. Naudojantis šia sistema, turėtų būti įmanoma objektyviau atsižvelgti į turimą uždavinį bei taip palengvinti mokymą bei gabumų įvertinimą.

Kuriamo projekto tikslas – išsiaiškinti, kokias galima naudoti technologijas, metodus bei algoritmus, ir atradus efektyvesnius metodus sukurti sistemą, kuri gavus programavimo uždavinį, perskaitytų

gautus paveikslėlius bei tekstą ir bandytų atspėti, kaip sudėtinga būtų kitam vartotojui teisingai jį atlikti, jei prieš tai jis jo nebuvo matęs.

1. Teorinė dalis

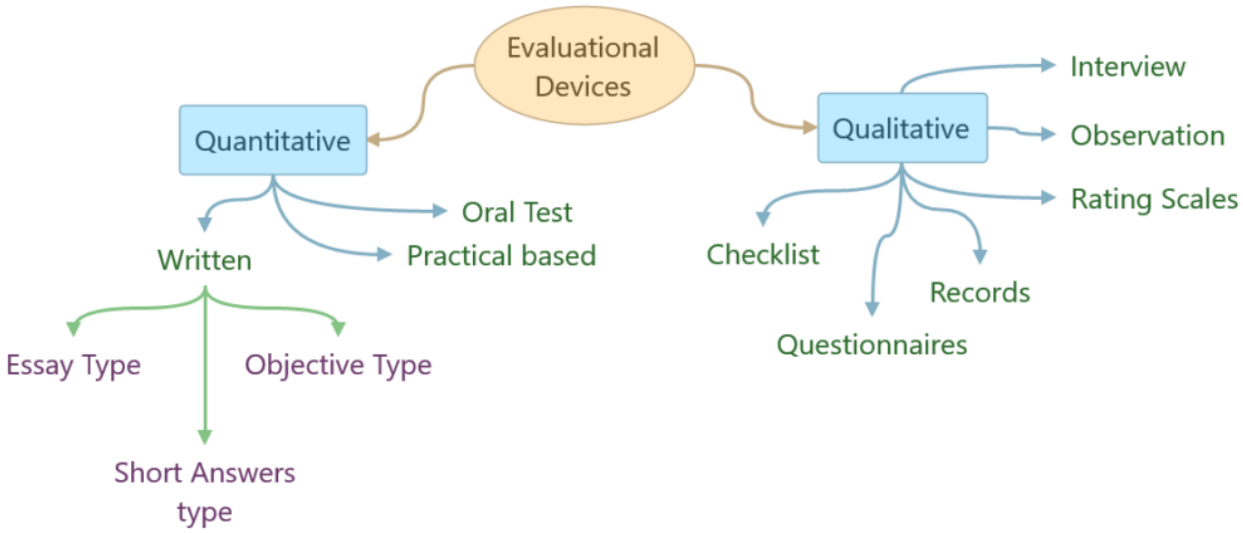
Šis skyrius skirtas aprašyti įvairius aspektus darbo, kuriame bus bandoma pasitelkti atliktais bandymais, algoritmais ir t.t.

1.1. Žinių vertinimo metodikos

Vertinimas gali būti:

- Diagnostinis – atliekamas norint pagerinti mokymosi procesą.
- Formuojamasis – nuolatinis vertinimas, padedantis numatyti mokymosi perspektyvas.
- Apibendrinamasis – vertinimas naudojamas baigus apmokymą.

Taip pat vertinimai gali būti išskaidomi į kokybinius bei kiekybinius [1]:



1 pav. Įvairūs dažnai naudojami įvertinimo būdai

1.1.1. Uždavinių sudarymo principai

Uždavinys skirtas įvertinti tą uždavinį atliekančio žinias bei gebėjimus. Užduotims paruošti keliami kai kurie reikalavimai:

- Tam tikros srities uždavinį dažniausiai sudaro ilgesnis uždavinio aprašymas, kurio atsakymas turi būti kokio nors tipo rezultatas.
- Galima priskirti papildomus apribojimus, pvz. laiką arba resursus, kuriuos galutinė programa turi naudoti.
- Uždavinys analizuojamas pagal jo atitikimą į norimą atsakymą.
- Klausimai turi būti aiškūs, logiški ir teisingi pagal dalykinę sritį.
- Klausimais tikrinama ne pastabumas arba reakcija, o studento žinios.

Klausimų sudarymui reikia mąstymą skatinančios medžiagos (iliustracijų, lentelių su duomenimis, diagramų, žemėlapių, t.t.), ir tą medžiagą reikia išanalizuoti tiek iš atliekančio, tiek iš įvertinančio pozicijos. Jie turi būti vienodo sunkumo abejoms lytims ir turėtų neįžeisti žmonių su kitokiais įsitikinimais, pomėgiais arba kultūra. Yra du pagrindiniai klausimų tipai:

- Uždarojo atsakymo klausimai – klausimai, kuriems atsakymai jau parašyti, studentui reikia išrinkti teisingą variantą. Pavyzdžiai:
    - Pasirenkamojo atsakymo klausimai
    - Teisingo / neteisingo atsakymo klausimai
    - Susiejimo klausimai (susieti atsakymus rodyklėmis)
    - ...

Uždarojo atsakymo klausimai naudojami matuojant studento suvokimą bei pritaikymą. Jei punkte naudojama neigiama formuluotė, ją reikėtų pabrėžti pabraukiant arba **išryškinant** raides ir neiginį nukeliant kaip galima toliau į formuluotės pabaigą. Jie dažnai garantuoja objektyvų vertinimą, tačiau didėja spėjimo tikimybė. Susiejimo klausimai tinka, kai reikia gebėti susieti du dalykus (įvykius ir datas, autorius ir darbus, t.t.), tačiau gali reikalauti daug laiko.

- Atviro atsakymo klausimai – klausimai, kuriems atsakymai neparašyti, studentui reikia teisingai parašyti atsakymą: Pavyzdžiai:
    - Trumpo atsakymo klausimai
    - Struktūriniai klausimai
    - Esė
    - Programavimo uždaviniai
    - ...

Atviro atsakymo klausimai naudojami tiksliau matuojant studento žinias. Jie garantuoja, kad atspėti nežinant atsakymo neįmanoma, tačiau reikia asmeniškai tikrinti atsakymus ir yra tikimybė neobjektyvaus įvertinimo. [2]

1.2. Buvę bandymai

Šiuo metu bandymų būtent tokio pobūdžio problemai spręsti nebuvo, tačiau yra keli susiję rezultatai panašiose srityse. Vienas svarbus pavyzdys būtų bandymas įvertinti įvairių klausimų sudėtingumą skaitymo uždaviniuose [3], kuris pritaikė specifinę specializuotą struktūrą, vadinamą „Testams pritaikytas dėmesiu paremtas konvoliucinis neuroninis tinklas“ (angl. *Test-aware Attention-based Convolutional Neural Network,* TACNN). Pagal aprašytus rezultatus, atliktas bandymas buvo sėkmingas – rezultatai geresni už kitus dažniausiai žinomus modelius. Taip pat buvo atliktas testavimas tarp ekspertų ir šios sistemos – rezultatai rodo, kad pritaikytas dirbtinis intelektas dažnai tiksliau įvertina gautus uždavinius su priskirta sudėtingumo informacija, lyginant su ekspertais.

Taip pat kitas bandymas buvo pritaikyti hibridinį dirbtinį intelektą (angl. *hybrid AI*) [4] įvertinti uždavinių sudėtingumą. Sistemai pradėti, mokytojas arba dėstytojas atlieka priskirto eksperto rolę. Tai atlikus, bet kokie atsiliepimai iš turimų studentų būna išsaugomi bei vėliau pritaikomi genetiniam algoritmui. Iš šio bandymo metu surinktų duomenų, rodoma kad dažnai pradinis eksperto įvestas variantas pagal studentus yra netikslus. Taip pat šis bandymas nėra teisingai pritaikytas – sistema naudoja tik vieną taisyklių įvertinimą, taigi dažnai gali įvykti, kad gautas sunkumo įvertinimas pasikeičia per daug arba per mažai, nes pridedant taisyklę panaikinama buvusi. Taip pat, jei pridėtos kelios taisyklės vienam sudėtingumo lygmeniui, sistema negali teisingai įvertinti, kurią taisyklę turėtų pritaikyti. Straipsnyje aprašyta, kad šiuo metu taisyklė iš kelių yra pasirenkama atsitiktiniu būdu.

Dalinai susijęs eksperimentas – bandymas įveikti MAB (Multi-Armed Bandit) problemą. MAB problema – situacija, kai turima įvairius galimus pasirinkimus ir norima atrasti kokį nors kiekį geriausių rezultatų, tačiau gaunami rezultatai pradžioje neaiškūs. Buvo bandomi įvairūs metodai tai atlikti:

- Tiesioginis (angl. *Direct*) – bandomas toks pats pavyzdžių kiekis kiekvienam variantui ir parenkami variantai su aukščiausiais vidurkiais.
- Dalinimo (angl. *Halving*) – bandymai padalinami į kokį nors „raundų“ kiekį. Po kiekvieno „raundo“, atmetama pusė žemiausių vidurkių. Tai kartojama kiekvienam „raundui“, išskyrus paskutiniam, kuriame grąžinamas norimas kiekis variantų.
- Nuoseklus priėmimas ir atmetimas (angl. *Successive Accepts and Rejects,* SAR) – bandymai padalinami į nenustatytą „raundų“ kiekį. Kiekvieną „raundą“, priimami geriausi ir atmetami prasčiausi variantai. Sustabdoma, kai priimta pakankamai variantų.
- Optimalus skaičiavimo biudžeto paskirstymas (angl. *Optimal Computing Budget Allocation,* OCBA) – veikia panašiai kaip SAR bei dalinimo metodai, tačiau šis metodas taip pat atkreipia dėmesį į gautų rezultatų įvairovę. Bandymai padalinami į kokį nors „raundų“ kiekį. Pirmą „raundą“ lygiai įvertinami visi, kitais „raundais“ padalinami bandymai pagal gautą koeficientą tarp vidurkio ir nuokrypio.

Taip pat yra papildomų metodų, kurie čia neaprašyti. [5]

Eksperimente nurodomi bandymai teisingai įvertinti visų metodų efektyvumą. Vienintelė problema su šiuo eksperimentu yra tai, kad pritaikymui atviro tipo programavimo uždaviniams reikėtų žymiai jį pakeisti, tačiau galima įžvelgti kelis dalykus su šiuo eksperimentu dėl galimo savarankiško mokymosi modeliui.

Daugiau bandytų variantų rašymo metu nebuvo atrasta.

1.3. Algoritmai

Įvairūs anksčiau kurti algoritmai gali būti pritaikyti šio tipo darbui. Kadangi naudojamas metodas (detaliau paaiškintas kitoje sekcijoje) naudoja ir tekstą, ir paveikslėlius, sukurti du skirtingi poskyriai, skirti aprašyti labiau žinomus algoritmus susijusius su jų įvertinimu. Taip pat pridėtas papildomas poskyris, kuriame bus aprašomi būdai sujungti modulius į vieną.

1.3.1. Tekstu paremto įvertinimo algoritmai

Natūralios kalbos tvarkymas (angl. *Natural Language Processing, NLP*) pirma reikalauja išankstinio pertvarkymo tekstui [6]. Čia minima maža dalis galimų būdų atlikti išankstinį pertvarkymą.

- **Tokenizacija** (angl. *Tokenization*) – procesas, kuro metu didelė teksto dalis išskaidoma ir klasifikuojama į mažesnes, kompiuteriui lengviau suprantamas dalis. Dažniausiai pritaikomi tokenizavimo būdai – išskaidyti tekstą pagal žodžius arba pagal sakinius. [7]

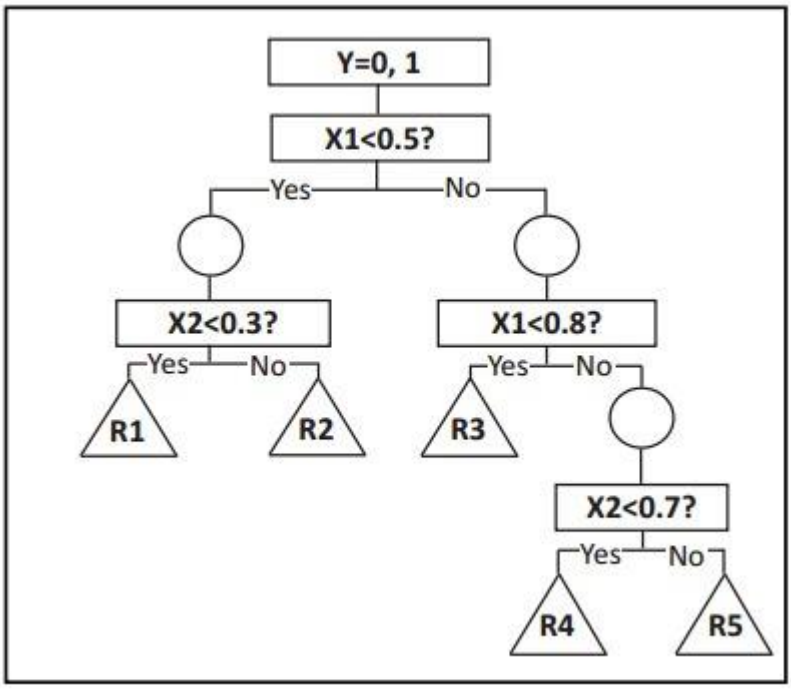
- **Normalizacija** (angl. *Normalization*) – metodas, skirtas patobulinti algoritmo efektyvumą panaikinant arba sugrupuojant į bendrą grupę pagal bendrus bruožus. Pavyzdžiui, **kamieno atrinkimas** (angl. *stemming*) sutrumpina žodžius į šakninę formą, ir **lematizacija** (angl. *lemmatization*) skirtas žodžių variantines formas sugrupuoti kartu. [8]

- **Žodžių „maišas“** (angl. *bag-of-words*) – modelis, kuris surenka visas žodžių leksemas iš teksto ir juos sudeda į vieną „maišą“ – formatą, kuris nekreipia dėmesio į žodžių tvarką ar gramatiką, tačiau išsaugo, kiek kartų kiekvienas žodis buvo pasirodęs sistemai skirtuose mokymo duomenyse, kad galėtų įvertinti, kaip sistema turėtų tai panaudoti. [9]

- **Term Frequency-Inverse Document Frequency** (TF-IDF) – dažnai automatinio teksto apdorojimo metu naudojama statistika, įvertinanti kiek svarbus tekste arba teksto dalyje naudojamas žodis yra dokumente. Ši statistika padidėja pagal pasirodžiusio žodžio dabartiniame dokumente dažnį, ir mažėja pagal pasirodžiusio žodžio praeityje ištirtuose dokumentuose dažnį. [10]

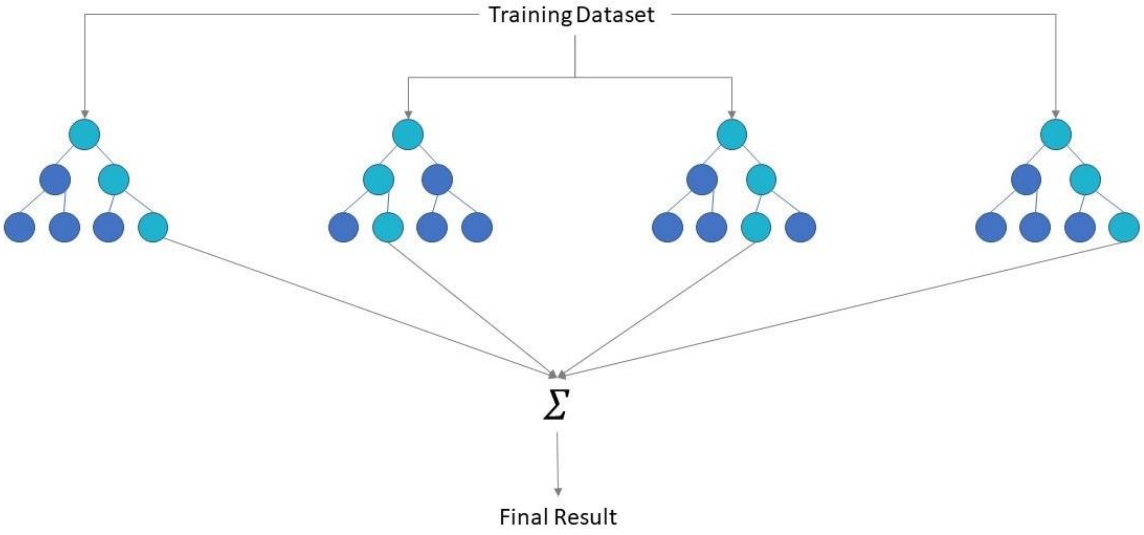
Pačiam tekstu paremtam įvertinimui galima naudoti įvairius algoritmus. Čia bus aprašoma dalis šioje sistemoje naudojamų algoritmų.

- **Pasirinkimų medis** (angl. *decision tree*) [11] – paprastas metodas klasifikacijai, kuris naudoja kelis „true/false“ klausimus suformuoti galutinį rezultatą. Pasirinkimo medžio klasifikatorius mašininio mokymo kontekste šio tipo „medį“ sukuria naudojant mokymo duomenis, kad būtų galima įvertinti, kokios įvestys turėtų būti naudojamos gauti tikslius rezultatus. Šis algoritmas dažnai naudojamas dėl jo paprastumo bei efektyvumo. Taip pat yra keli dažnai pritaikomi būdai patobulinti sistemą (šie būdai aprašyti modulių sujungimo poskyryje). Toliau parodytas paprastas pasirinkimų medžio pavyzdys. Kai modulis suformuotas, galima modulį apkarpyti (angl. *pruning*), kad būtų galima sumažinti nereikalingų šakų kiekį ir taip pagreitinti bei supaprastinti modulį.



2 pav. Pavyzdinis pasirinkimų medis, sukurtas pagal kintamąjį Y.

- **Atsitiktinis miškas** (angl. *random forest*) [12] – pasirinkimų medžių pritaikymo metodas. Šiam modeliui sukuriami keli skirtingi pasirinkimų medžiai. Tada kiekvienam medžiui sukuriama nauja duomenų aibė, kurioje kai kurie duomenys pakeičiami kitais, kad visi medžiai būtų apmokyti skirtingai. Norint iš atsitiktinio miško gauti rezultatą, duomenys pateikiami visiems medžiams, tada iš visų gautų rezultatų atrenkamas dažniausiai gaunamas rezultatas. Atsitiktinio miško modelis buvo sukurtas pataisyti su pasirinkimų medžiais dažnai iškylančią problemą – juos apmokant labai dažnai per daug pritampa prie suteiktų duomenų (angl. *overfitting*).

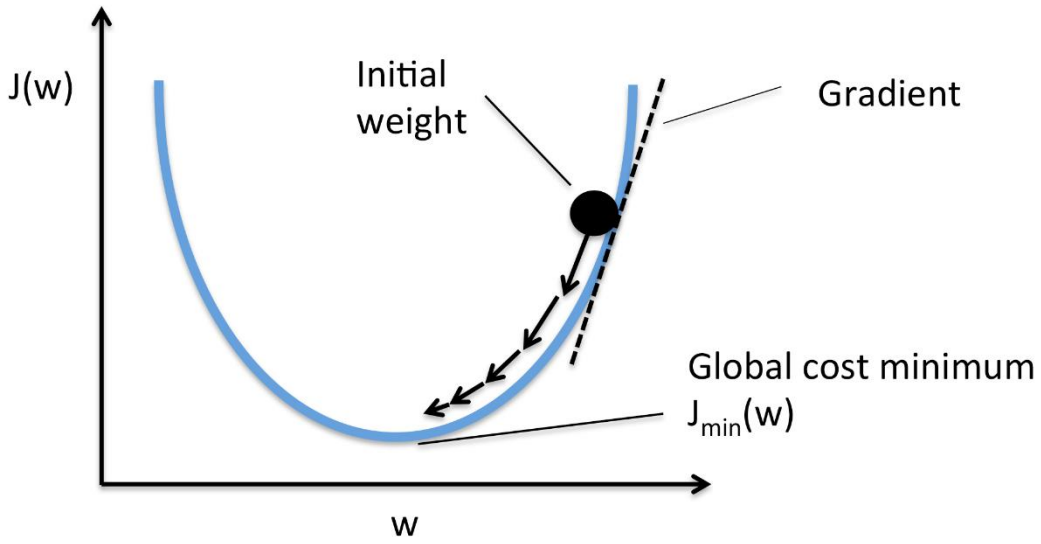


3 pav. Atsitiktinio miško modulio pavaizdavimas

- **LightGBM** (Light Gradient Boosting Machine) [13] – gradientų tobulinimo pasirinkimų medis (angl. *gradient boosting decision tree, GBDT*), sukurtas Microsoft. Ši realizacija pritaiko vienašalį mėginių ėmimą (angl. *Gradient-based One-Side Sampling, GOSS*), kuris leidžia surinkti tikslią informaciją su mažiau turimų duomenų, bei išskirtinių požymių surinkimas (angl. *Exclusive Feature Bundling, EFB*), kuris jungia duomenis, kurie negali abu turėti tam tikrų reikšmių vienu metu, Šias technologijas pritaikius sistema žymiai pagreitėja bei išlaiko apytiksliai tą patį rezultatą.

- Ribotos atminties Broyden–Fletcher–Goldfarb–Shanno algoritmas (angl. *Limited-memory Broyden–Fletcher–Goldfarb–Shanno*), dažniausiai vadinamas **L-BFGS** [14] – metodas, skirtas įvertinti Broyden–Fletcher–Goldfarb–Shanno (BFGS) algoritmą, kuris padeda įvertinti kitą reikšmę naudojantis vidutine reikšme bei kitais veiksniais.

- **Stochastinis gradientinis nusileidimas** (angl. *Stochastic gradient descent*. SGD) – iteracinis metodas, kuris bando atliekant kelis žingsnius atrasti „žemiausią“ tašką grafike. Skiriasi nuo paprasto gradientinio nusileidimo modeliu tuo, kad naudoja tik dalį viso grafiko, kad būtų galima greičiau atrasti tą tašką. Vienas iš dažniausiai pritaikomų metodų mašininiam mokymui.



4 pav. Stochastinio gradientinio nusileidimo pavaizdavimas

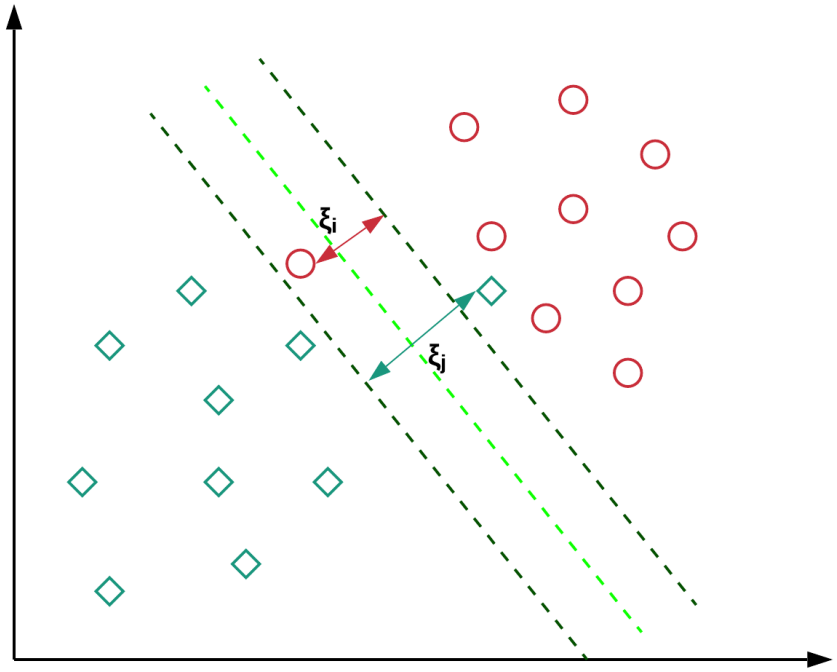
-

- **Vidutinis perceptronas** (angl. *Averaged perceptron*) [15] – vienas iš labiau žinomų algoritmų, ir pradinis algoritmas, iš kurio vėliau išsivystė pradinė neuroninių tinklų versija. Šis modelis priima duomenis ir juos pakeičia naudojant paprastą algoritmą:

$$a = \left[\sum_{d=1}^{D} w_d x_d\right] + b$$

- **Tiesinis atraminių vektorių klasifikatorius** (angl. *Linear Support-Vector Machine*) [16] – modelis, kuris skirtas tik atlikti dvejų rūšių klasifikaciją. Šis modelis veikia taip – suteikiami duomenys apie „taškus“ ir klasę, kuriai jie priklauso. Modelis tada pagal gautus duomenis

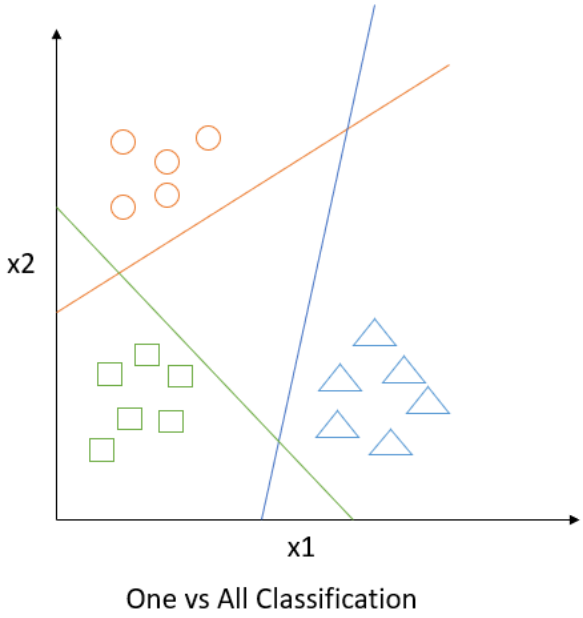
ieško būdo atskirti duomenis taip, kad būtų kuo įmanoma didesnis atstumas tarp dvejų tipų. Lygtinai greitas algoritmas, tačiau gali sutrikti, jei išdėliojus taškus lieka neaiškumų su duomenimis.



5 pav. Tiesinio atraminių vektorių klasifikatoriaus veikimo atvaizdavimas

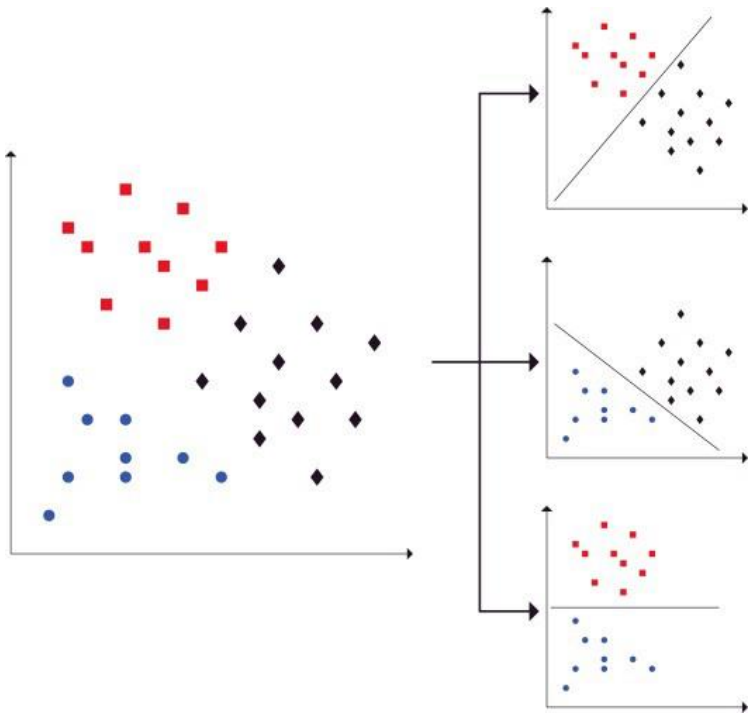
Kai kurie klasifikavimo metodai sukurti tik dvejų rūšių klasifikacijai (angl. *binary classification*). Jei norima pakeisti daugelio rūšių klasifikacija (angl. *multiclass classification*), galima tai atlikti vienu iš dviejų būdų [17]:

- **„Vienas prieš visus“** (angl. *One-vs-All*) metodas sukuria n skirtingų modelio versijų, kur n yra skirtingų klasifikacijų kiekis. Visi moduliai atlieka dvejų rūšių klasifikaciją – tikrinama, ar priklauso kuriai nors grupei, ar ne. Galutinis modulis grąžina labiausiai atsakymą atitinkantį rezultatą. Šis metodas greitesnis bei reikalauja mažiau atminties.



6 pav. „Vienas prieš visus“ veikimo pavaizdavimas

- **„Vienas prieš vieną“** (angl. *One-vs-One*) metodas sukuria skirtingą modelį kiekvienam galimam variantui. Kai visi moduliai atlieka savo darbą, teisingas atsakymas išrenkamas susumuojant ir gaunant vidurkį visų susijusių modulių bei grąžinamas aukščiausias rezultatas. Šis metodas sudėtingesnis, lėtesnis bei reikalauja daugiau atminties, tačiau kai kuriais atvejais būna tikslesnis nei „Vienas prieš visus.“



7 pav. „Vienas prieš vieną“ veikimo pavaizdavimas

Taip pat yra keli metodai, kurie leidžia atlikti daugelio rūšių klasifikaciją tekstui nenaudojant vieno iš anksčiau aprašytų metodų:

- Neuroninis tinklas, kuris detaliau bus aprašomas tolesnėje dalyje.

- k-arčiausių kaimynų metodas (angl. k-nearest neighbors) – Algoritmas, kuris išdėlioja visus turimus duomenis į grafiką ir tada apskaičiuoja atstumą tarp taškų. Tada pagal gautus atstumus surenkami *k* arčiausių taškų. Šio tipo uždaviniui lygtinai sunku ši metodą pritaikyti be pakeitimų.

1.3.2. Paveikslėliais paremto įvertinimo algoritmai

Kompiuterinis regėjimas (angl. *Computer vision*) – bandymas sistemą atkurti taip, kai kompiuteris vaizdu susijusias problemas įvertina panašiai, kaip ir tikras asmuo. [18] Dažnai toks atpažinimas turi įvertinti specifinius dalykus, pvz. stiprumą.

Pagrindinis dalykas, skirtas paveikslėliais paremto įvertinimo algoritmams – neuroniniai tinklai, t.y. tinklas, sudarytas iš daugelio „neuronų“ sluoksnių.

Paveikslėlių klasifikacijai įvykdyti dažnai naudojamas **gilusis mokymas** (angl. *deep learning*) – mašininio mokymo forma, kuri pritaiko specialius neuroninius tinklus, kurie pasižymi didesniu tikslumu bandant atpažinti . Tai atlikti pritaikomas gilusis neuroninis tinklas (angl. *deep neural network*). Šio tipo neuroninio tinklo išskirtinė savybė yra tai, kad ji turi kelis paslėptus sluoksnius (angl. *hidden layer*), t.y. sluoksnius po įvesties sluoksnio, bet prieš išvesties sluoksnį, kurie atlieka dalines operacijas.

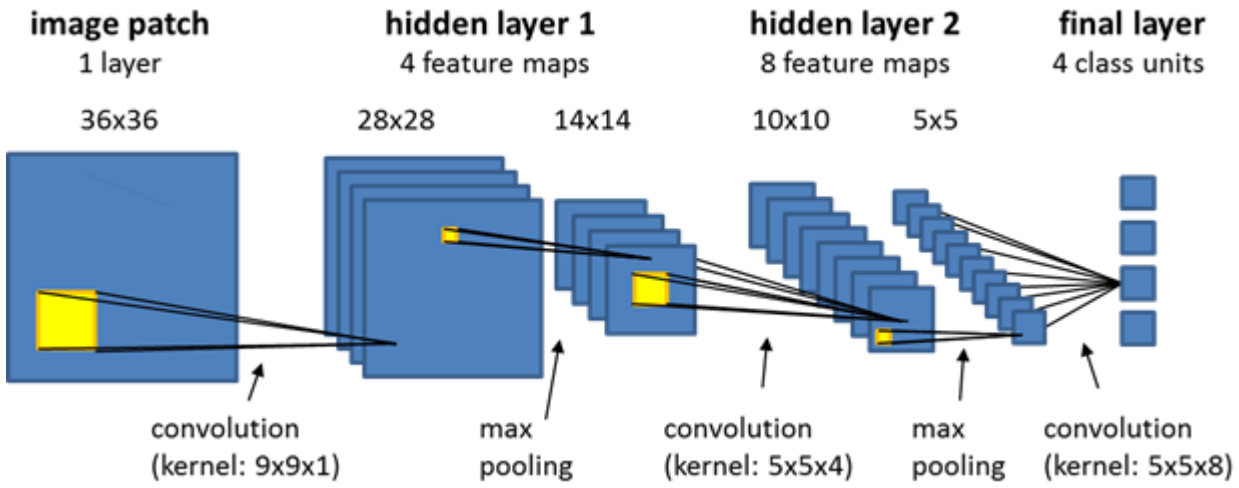
Savybių mokymasis (angl. *feature learning*) – giliajam mokymui pritaikomas metodas, kurio metu paveikslėliui išmokstama kaip atpažinti savybes (t.y. specifines išskirtines paveikslėlio savybes), ir pagal tas atrinktas savybes bandoma toliau kategorizuoti paveikslėlį, taip pasiekiant didesnį tikslumą.

Dažnai paveikslėlių atpažinimo ir kategorizacijos uždaviniams pritaikomas **konvoliucinis neuroninis tinklas** (ang. *Convolutional neural network*) – gilusis neuroninis tinklas, pasižymintis savybių mokymosi galimybe. [19] Šie neuroniniai tinklai pasitelkia tris pagrindinius neuronų sluoksnius:

- **Visiškai sujungti sluoksniai** (angl. *Fully-connected layers*) - dažnai naudojimas algoritmas. Šio sluoksnio pagrindinė charakteristika yra tai, kad visi sluoksnio neuronai yra visiškai sujungti su kito sluoksnio neuronais.

- **Konvoliuciniai sluoksniai** (angl. Convolutional layers) supaprastina gautus duomenis (dažniausiai paveikslėlius) ir išmoksta jų „savybes“, pagal kurias toliau esantys sluoksniai gali įvertinti. Šis sluoksnis kartu su toliau aprašytu apjungimo sluoksniu padeda išspręsti problemą su paveikslėlių apdorojimu neuroniniu tinklu - aukštos rezoliucijos nuotraukos reikalauja per daug įvesčių, kad įprastas neuroninis tinklas veiktų greitai ar efektyviai.

- **Apjungimo sloksnis** (angl. *Pooling layers*), kurio metu bandoma surinkti informaciją bei sumažinti gautą nuotrauką, kad būtų galima ją naudoti kituose sluoksniuose su mažiau neuronų. [20]



8 pav. Konvoliucinio neuroninio tinklo pavyzdys

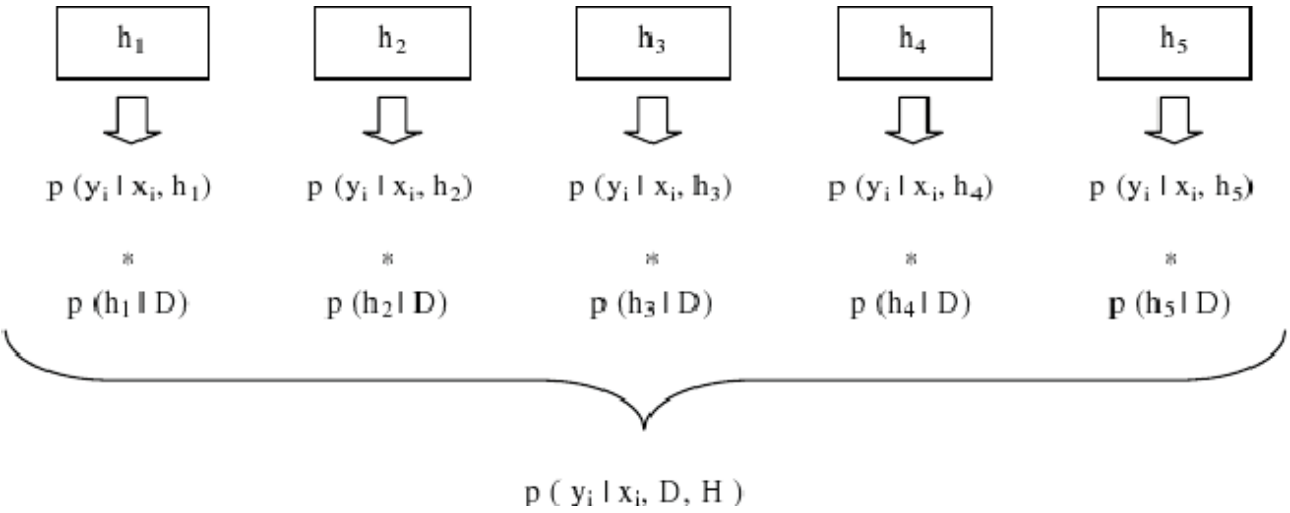
1.3.3. Modelių sujungimas

Yra keli skirtingi metodai, kurie praeityje buvo naudojami sujungti kelis modelius ir iš bendro modulio gauti vieną atsakymą. Dažnai visi sujungiami modeliai būna identiški, tačiau jiems apmokyti naudojami skirtingi duomenys. Šitoks sujungimas vadinamas „ansambliavimo mokymas“ (angl. *ensemble learning*). Dažniausiai naudojami ansambliavimo būdai yra šie:

- ***Bootstrap aggregating*** [21], kartais vadinamu ***bagging***, pirma apskaičiuojama sudėtinių modelių tikslumas. Tada jų rezultatai sudedami bei iš jų suskaičiuojamas vidurkis – modelis su aukščiausiu rezultatu grąžina savo rezultatą. Vienas iš „*bootstrap aggregating*“ pritaikymų – **atsitiktinis miškas** (angl. *random forest*). Sukuriamas greitai, tačiau tikslumas bei kokybė nukenčia.

- ***Bayesian Model Averaging*** [22] – metodas, labai panašus į anksčiau aprašytą „*bootstrap aggregating*“. Mokymas vyksta taip – duomenys atsitiktinai išdėliojami kitokia tvarka, išmokius bandoma gauti įvertinimą iš sistemos. Skirtumas tuo, kad kiekvienas modelis turi priskirtą svorį, taigi dėl to šis variantas būna tikslesnis, tačiau su šiuo modeliu iškyla nemažai atvejų, kai visas svoris sudedamas į vieną modelį.

- ***Bayesian Model Combination*** [23] – metodas, kurtas pagal anksčiau paminėtą *Bayesian Model Averaging*, ir skirtas ištaisyti jame atrastą matematinę klaidą. Vietoj to, kad būtų be priežasties bandomas kiekvienas ansamblyje esantis modelis individualiai, bandoma iš galimų ansamblių, taip pataisant svarbią *Bayesian Model Averaging* klaidą – šis modelis daug rečiau sudeda visą svorį į vieną modelį. Pareikalauja šiek tiek daugiau resursų, bet grąžina žymiai tikslesnius rezultatus.

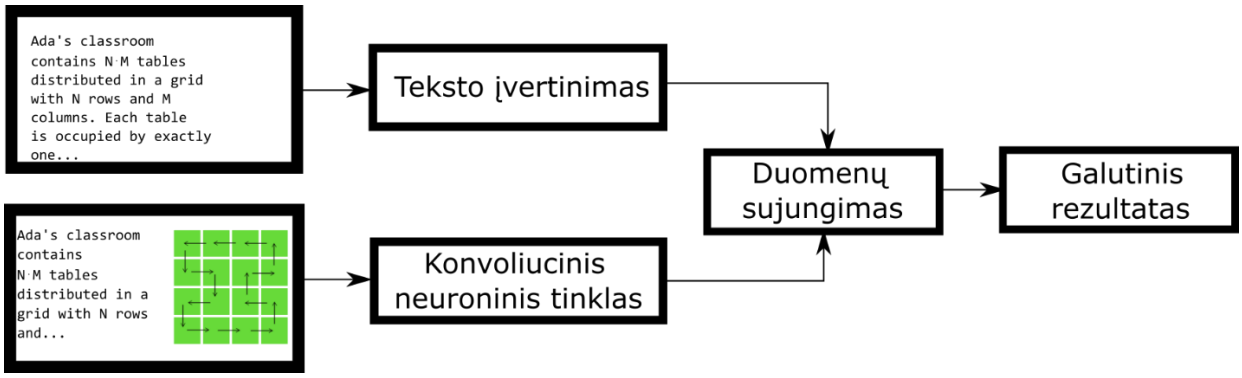


9 pav. „Bayesian Model Averaging“ pavaizdavimas

- **Modelių kibiras** (angl. *Bucket of models*) – ansambliavimo metodas, skirtas išrinkti geriausią galimą modelį kiekvienam atvejui. Jis veikia panašiai, kaip ir „*Bootstrap aggregating*“ – apjungiama daug modelių. Pagrindinis šio modelio skirtumas yra tai, kad galima sudėti skirtingus modelius, tačiau tai visvien tik atranda geriausią tam atvejui skirtą modelį, jų tikrai neapjungia.

Yra dar vienas būdas pritaikyti du skirtingus modelius sujungti į vieną – **dvišakiai neuroniniai tinklai** [24] (angl. *two-branch neural network*). Juos naudojant galima sujungti teksto bei paveikslėlių įvertinimo sistemas į vieną, siekiant gauti dar tikslesnius atsakymus iš mašininio mokymo.

Dvišakis neuroninis tinklas šiame kontekste pirma priima pradinius duomenis dviems neuroniniams tinklams – vienas paveikslėliams, kitas paveikslėliui, kuris naudojamas įvertinimui. Jie abu atlieka įprastus savo įvertinimus, ir atlikus tai gauti rezultatai sujungiami į vieną. Tada vienas neuronų sluoksnis panaudojamas iš tų rezultatų dar kartą įvertinti ir gauti galutinį rezultatą.



10 pav. Dvišakio neuroninio tinklo veikimo pavyzdys.

1.4. Duomenų rinkinys

Šiuo atveju duomenų rinkinys buvo asmeniškai surinktas, naudojantis įvairiomis svetainėmis, pvz. *CodeChef* bei *HackerRank*. Surinkta bei išdėliota apie 5000 skirtingų įrašų, kuriuose kiekvienas turi

sudėtingumo lygį (pradinis, lengvas, vidutiniškas, sunkus, ekspertams), uždavinio pavadinimas, aprašymas bei besikeičiantis paveikslėlių kiekis.

2. Projektinė dalis

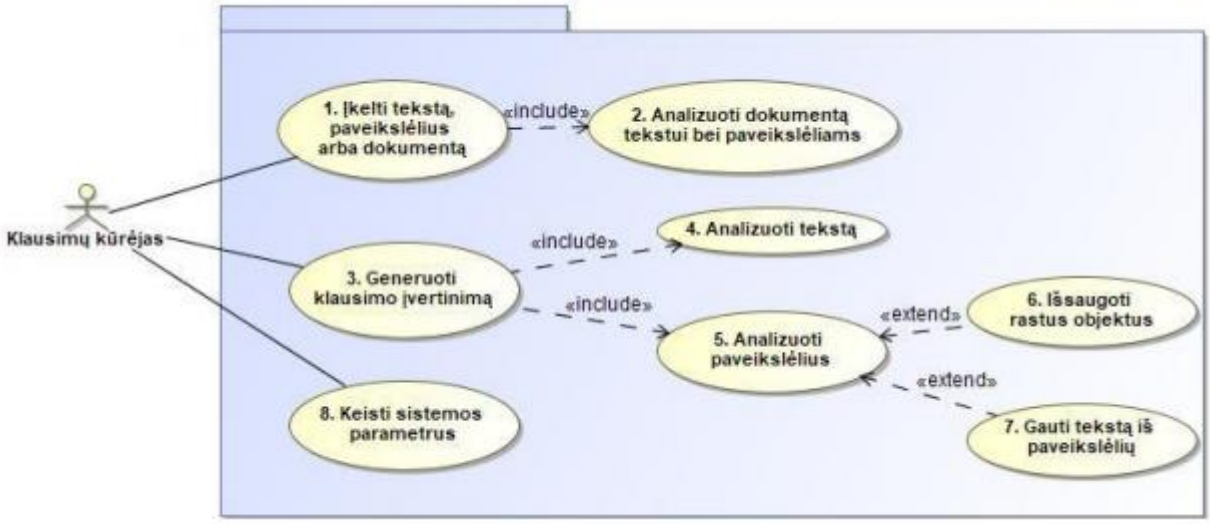
Šio tipo uždaviniui reikia įvertinti ir tekstą, ir paveikslėlius, taigi sukuriami modeliai, kurie atskirai juos įvertina. Taip pat sukurtas dvišakis neuroninis tinklas, kuris bendrai įvertina gautus atsakymus.

2.1. Reikalavimai ir architektūra

2.1.1. Sistemos tikslai

- Patobulinti bei supaprastinti sudėtingumo įvertinimą uždaviniams, kad būtų jų kūrėjams paprasčiau nutarti, kokio sudėtingumo jų sukurtas uždavinys
- Naudojantis sugeneruotais sudėtingumo įvertinimais, galima pritaikyti mokymo ar uždavinių įvertinimo srityje
- Galima pritaikyti, jei reikia sukurti specifinio lygio uždavinius
- Galima naudoti analizei, kaip reikėtų frazuoti bei kokius paveikslėlius naudoti, kad palengvinti darbą

2.1.2. Sistemos panaudojimo atvejai



11 pav. Panaudojimo atvejų diagrama

2.1.3. Nefunkciniai reikalavimai

- Turi būti lengva naudotis sistema
- Vartotojas neturėtų ilgai užtrukti ieškant funkcijos
- Sistema neturėtų trukdyti kitai programinei įrangai
- Turėtų greitai atlikti darbą
- Turi būti sukurta taip, kad ateityje būtų galima lengvai sistemą praplėsti

- Sistema negali pasiekti duomenų, kurie nebuvo suteikti vartotojo

2.1.4. Bendradarbiaujančios sistemos

Pilna sistema išskaidoma į penkias posistemes:

- Teksto perskaitymas – posistemė perskaito jai suteiktą tekstą, taip pat surasti raktinius žodžius bei įvertinti teksto sudėtingumą. Sujungta kartu su paveikslėlių atpažinimo posisteme.

- Paveikslėlių atpažinimas – posistemė analizuoja paveikslėlius ir bando rasti įvairias detales. Jas suradus, laikinai išsisaugo ir surinktomis savybėmis naudojantis įvertina paveikslėlio sudėtingumą.

- Duomenų iš dokumentų surinkimas – posistemė perskaito atitinkamo formato dokumentą, ir baigus perskaitymą, prideda tekstą bei paveikslėlius atitinkamoms posistemėms perskaityti.

- Uždavinio sudėtingumo įvertinimas – posistemė priima gautus duomenis iš teksto perskaitymo bei paveikslėlių atpažinimo posistemių ir jų duomenis išveda (sudėtingumo įvertinimas ir tikimybė kiekvienam sudėtingumo lygiui ir teksto, ir paveikslėlių sudėtingumo įvertinimui).

- Dvišakis atpažinimas – posistemė priima gautus duomenis iš teksto perskaitymo bei paveikslėlių atpažinimo posistemių ir jų duomenis panaudoja gauti galutinį rezultatą.

- Vartotojo sąsaja – ši posistemė apjungia visas kitas posistemes, taip pat parodyti reikiamą informaciją vartotojui bei jam leisti atlikti įvairius veiksmus.

2.2. Tekstu paremtas sudėtingumo įvertinimas

Tyrimui buvo bandoma naudoti įvairius teksto įvertinimo metodus – perceptronus, LightGBM, pasirinkimų medžius ir kitus – ir patikrinamas jų tikslumas. Detaliau šio įvertinimo rezultatai pavaizduoti toliau pavaizduotoje lentelėje. „MicroAccuracy“ – tikslumas pagal esančių duomenų kiekį, o „MacroAccuracy“ – tikslumas pagal kiekvienos galimos kategorijos duomenų tikslumą.

Išankstiniam duomenų pertvarkymui kiekvienam galimam galutiniam rezultatui priskiriamas „raktas“, t.y. identifikuojantis skaičius. Taip pat kitų įvesčių duomenys perskaitomi ir pakeičiami į n-gramas (n objektų sekas). Tai atliekama ir žodžiams, ir atskiriems simboliams. Gautos n-gramos toliau naudojamos įvertinti pradinius svorius, ir taip pat padeda įvertinti, kiek ir kaip žodžiai turėtų paveikti gautą rezultatą apie uždavinio sudėtingumą.

Kad būtų galima paprasčiau suprasti, „MicroAccuracy“ bei „MacroAccuracy“ dalijasi pradine formule:

$$P_r = \frac{TP}{TP + FP}$$

Šioje formulėje, bei kitose šiame skyriuje aprašytose formulėse, TP reiškia teisingų atsakymų kiekį (*TP – True Positive*), o FP reiškia situacijų, kuriose gautas klaidingas rezultatas, kiekį (*FP – False Positive*)

Mikro-vidurkio tikslumas (angl. *Micro-average Accuracy, MicroAccuracy*) sudeda visus gautus rezultatus prieš atliekant likusį skaičiavimą. Pavyzdžiui, jei būtų trys galimi atsakymai, mikro-vidurkio tikslumas įvertinamas taip:

$$P_{r_{micro}} = \frac{TP1 + TP2 + TP3}{TP1 + TP2 + TP3 + FP1 + FP2 + FP3}$$

Makro-vidurkio tikslumas (angl. *Macro-average Accuracy, MacroAccuracy*) apskaičiuoja kitaip – kiekvienam galimam rezultatui pirma apskaičiuojamas vidurkis pagal anksčiau parašytą formulę, tada visi gauti rezultatai sudedami ir padalinami iš esančio galimų variantų kiekio. Pavyzdžiui, naudojant tą patį pavyzdį, su trimis galimais atsakymais makro-vidurkio tikslumas apskaičiuojamas taip:

$$P_{r_{macro}} = \frac{TP1}{TP1 + FP1} + \frac{TP2}{TP2 + FP2} + \frac{TP3}{TP3 + FP3}$$

Kitaip pasakius, mikro-vidurkio tikslumas skirtas įvertinti tikslumą, o makro-vidurkio tikslumas nurodo tikslumą nekreipiant dėmesio į duomenų kiekį kiekvienai klasei. Toliau esančioje lentelėje aprašomi gauti tikslumai sudėtingumui. Šiuo atveju mikro-vidurkio tikslumas ir makro-vidurkio tikslumas daug nesiskiria – tuo norima parodyti, kad nors ir duomenų kai kuriems sudėtingumo įvertinimams yra daugiau, tai žymiai nepaveikia daugumos pristatomų modelių galutinių rezultatų.

1 lentelė. Tekstu paremto sudėtingumo įvertinimo tikslumas su įvairiais modeliais

Modelio pavadinimas	**MicroAccuracy**	**MacroAccuracy**
Random Forest	0.718	0.725
Decision Tree	0.711	0.712
LightGBM	0.701	0.708
Logistic Regression (L-BFGS)	0.660	0.657
Maximum Entropy (SDCA)	0.639	0.661
Stochastic Gradient Descent	0.620	0.623
Averaged Perceptron	0.609	0.641
Linear Support-Vector Machine	0.583	0.597
SymbolicSgdLogisticRegression	0.545	0.584
Maximum Entropy (L-BFGS)	0.458	0.319

2.3. Paveikslėliais paremtas sudėtingumo įvertinimas

Daugeliu atveju paveikslėlių skaičius uždavinyje neaiškus – gali būti daugiau arba mažiau, arba gali visai jų nebūti, tačiau sistema turi juos įvertinti identiškai. Tai atlikti kiekvienam uždaviniui buvo sukurtas vienas bendras paveikslėlis, kuriame sudėti tekstas ir paveikslėliai. Apskaičiuojamas reikiama galutinio paveikslėlio rezoliucija, sukurtas teksto atvaizdavimas iš anksto nustatytu stiliumi, o paveikslėliai sudėti originaliai uždavinyje buvusiose vietose. Toliau pavaizduotas sugeneruoto paveikslėlio pavyzdys.


Ada School

Read problems statements in Hindi, Mandarin Chinese, Russian, Vietnamese and Bengali as well.
Ada's classroom contains
N · M
tables distributed in a grid with
N
rows and
M
columns. Each table is occupied by exactly one student.
Before starting the class, the teacher decided to shuffle the students a bit. After the shuffling, each table should be occupied by exactly one student again. In addition, each student should occupy a table that is adjacent to that student's original table, i.e. immediately to the left, right, top or bottom of that table.
Is it possible for the students to shuffle while satisfying all conditions of the teacher?
Input
The first line of the input contains a single integer
T
denoting the number of test cases. The description of
T
test cases follows.
The first and only line of each test case contains two space-separated integers
N
and
M
.
Output
For each test case, print a single line containing the string "YES" if it is possible to satisfy the conditions of the teacher or "NO" otherwise (without quotes).
Constraints
1≤T≤5,000
2≤N,M≤50
Example Input
2
3 3
4 4
Example Output
NO
YES
Explanation
Example case 2: The arrows in the following image depict how the students moved.
All submissions for this problem are available.
Author: 7★ alei
Editorial: https://discuss.codechef.com/problems/ADASCOOL
Tags: alei, conditional-statement, makicos, simple
Date Added: 31-08-2018
Time Limit: 1 secs

12 pav. Sugeneruoto paveikslėlio pavyzdys

Kai paveikslėliai sugeneruoti, jie panaudojami kaip duomenys giliajam neuroniniui tinklui, kuris įvertina jį. Buvo panaudoti keli dažnai naudojami modeliai – jų pavadinimai ir gautas tikslumas aprašyti toliau esančioje lentelėje. Patiems modeliams trūksta tikslumo – daugiausia pasiekiama tik 43%.

2.4. Tekstu paremto ir paveikslėliais paremto įvertinimo sujungimas

Norint patobulinti gautą sudėtingumo įvertinimo tikslumą, buvo bandyta pritaikyti dvišakį neuroninį tinklą. Šiuo atveju pirma naudojami du atskiri modeliai – vienas tekstui, vienas paveikslėliams. Jie įprastai įvertina duomenis, tada gauti rezultatai (gautas įvertinimas ir visos tikimybės, kad sudėtingumas būtų lygus tam tikram variantui) sujungti į vieną ir įvertinami iš naujo. Figūra, pavaizduojanti sukurto dvišakio modulio veikimą, buvo pavaizduota 2 pav. Iš naujo įvertinant šiuo atveju naudojamas vieno sluoksnio neuroninis tinklas.

Tekstu paremtas modelis turi daugiausiai apie 71% tikslumą, o paveikslėliais paremtas modelis – tik apie 40%. Naudojantis dvišakiu modeliu, gaunamas apie 80-90% tikslumas – didesnis už abu prieš tai buvusius modelius.

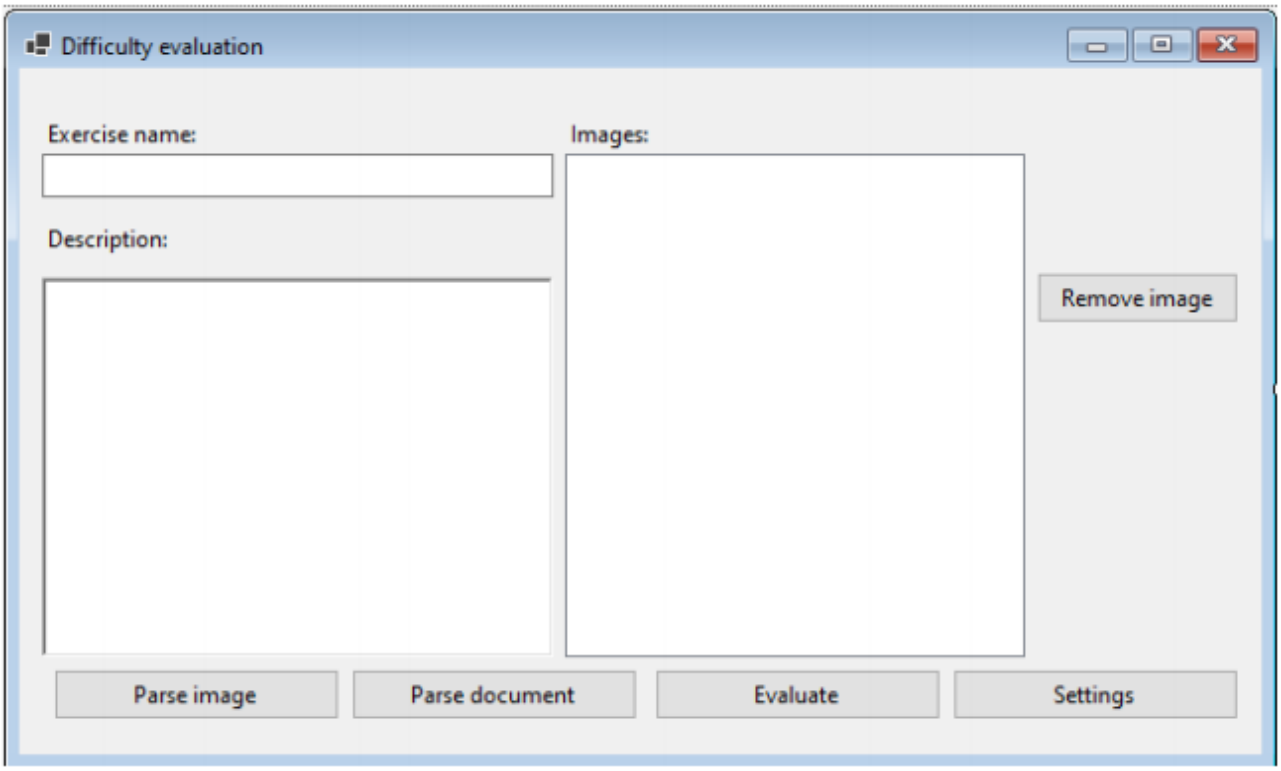
2.5. Vartotojo sąsaja

Vartotojo sąsaja susidaro iš šių dalių:

- Įvesties langas
- Nustatymų langas
- Galutinis rezultatas

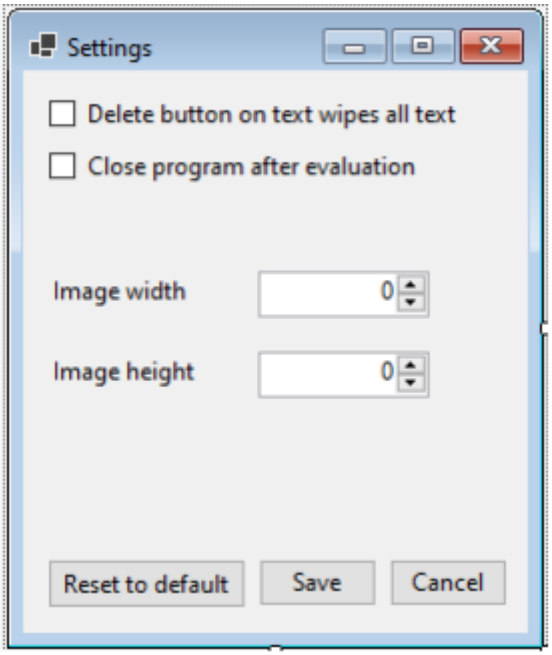
Įvesties langą sudaro teksto įvedimas bei keli mygtukai:

- Parse image – leidžia įkelti paveikslėlį, kuris bus išanalizuojamas surasti sudėtingumo lygį.
- Parse document – leidžia įkelti dokumentą, kurį nuskaičius sukels duomenis į programą.
- Evaluate – nuskaito suvestus duomenis ir bando juos išanalizuoti bei surasti sudėtingumo lygį.
- Settings – atidaro nustatymų langą



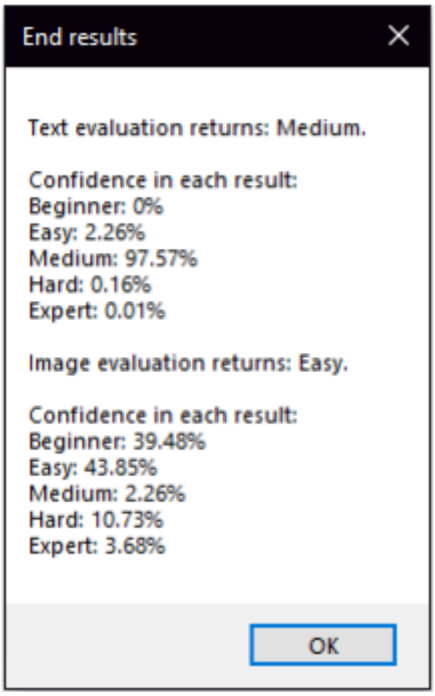
13 pav. Sudėtingumo įvertinimo įvesties lango pavyzdys

Nustatymų langas turi visus įvestus nustatymus, o galutinio rezultato langas išveda sudėtingumą bei tikimybes visų galimų rezultatų.



14 pav. Sudėtingumo įvertinimo nustatymų lango pavyzdys

Galutinis rezultatas grąžina paprastą langą, kuriame nurodomi gauti rezultatai.



15 pav. Rezultatų grąžinimo pavyzdys

3. Tyrimo dalis

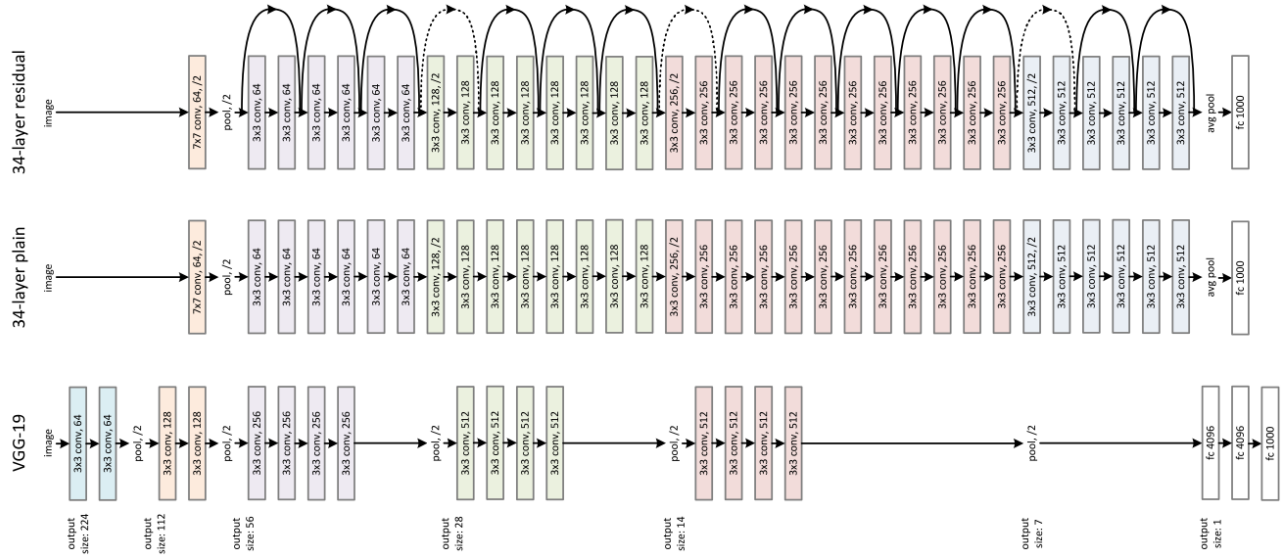
Buvo iškeltos kelios hipotezės:

3.1. Iš anksto apmokytų paveikslėlių modelių pritaikymas

– Galima naudoti skirtingus iš anksto apmokytus paveikslėlių klasifikacijos modelius ir iš to gauti kitokius rezultatus

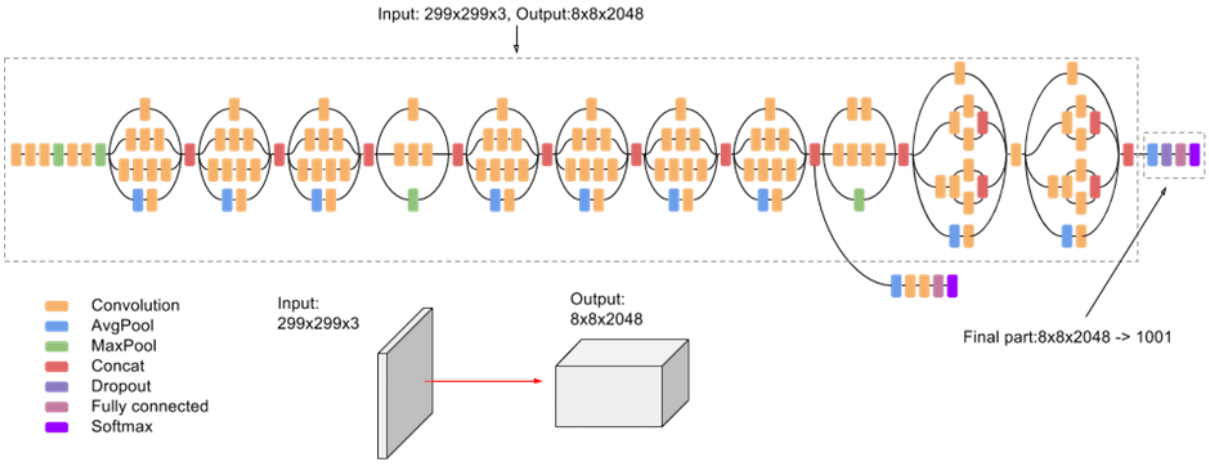
Šią hipotezę pilnai ištirti reikia pasitelkti keliais modeliais. Buvo atrasti šie modeliai:

1. **ResNet50** – ResNet architektūra, kuri turi vieną „average pool“ sluoksnį, 48 konvoliucinius sluoksnius ir 1 „MaxPool“ sluoksnį.
2. **ResNet101** – ResNet architektūra, kuri turi vieną „average pool“ sluoksnį, 99 konvoliucinius sluoksnius ir 1 „MaxPool“ sluoksnį. Lyginant su ResNet50 modeliu, dažniau pasiekia teisingus rezultatus įprastai klasifikacijai, tačiau daugelyje atvejų dėl dvigubai daugiau konvoliucinių sluoksnių nukenčia greitaveika.



16 pav. ResNet architektūra

3. **MobileNet V2** – speciali architektūra, kuri pritaikyta efektyviau veikti ant mobilių prietaisų, o ne stacionarių. Turi konvoliucinį sluoksnį su 32 filtrais ir 19 „išlikusios kliūties“ (angl. *residual bottleneck*) sluoksnių.
4. **Inception V3** – 42 sluoksnių architektūra, skirta supaprastinti paveikslėlių atpažinimą įvairių konvoliucijų tipais ir mažinimais.



17 pav. Inception V3 architektūra

Šie modeliai įprastai sugeba gauti nemažą tikslumą, tačiau regis šiuo atveju nepasiteisino – didžiausią tikslumą gavo ResNet101 modelis, bet jis pasiekė tik 43% tikslumą. Spėjama, kad tai įvyko dėl nesutapimo su pateiktu uždaviniu.

2 lentelė. Paveikslėliais paremto sudėtingumo įvertinimo tikslumas su įvairiais modeliais

Modelio pavadinimas	**MicroAccuracy**	**MacroAccuracy**
ResNet50	0.403	0.343
ResNet101	0.436	0.385
MobileNet V2	0.297	0.239
Inception V3	0.339	0.258

3.2. Dvišakio modelio rezultatai

– Galima naudoti skirtingus modelius dvišakiui modeliui.

Ši hipotezė buvo įvertinta keliais galimais variantais. Buvo panaudojami keturi įprasti variantai tekstui:

- Perceptronas (Averaged Perceptron)
- Pasirinkimų medis (Decision Tree)
- Linear Support-Vector Machine
- Logistic Regression (L-BFGS)

Taip pat buvo pritaikyti keturi anksčiau paminėti modeliai paveikslėliams:

- ResNet50
- ResNet101
- MobileNet V2
- Inception V3

Tekstu paremtas modelis turi daugiausiai apie 71% tikslumą, o paveikslėliais paremtas modelis – tik apie 40%. Naudojantis dvišakiu modeliu, gaunamas apie 81-94% tikslumas – didesnis už abu prieš tai buvusius modelius. Toliau esančioje lentelėje aprašyti keli iš tiksliausių modelių.

3 lentelė. Dvišakio sudėtingumo įvertinimo modelio tikslumas su įvairiais pradiniais modeliais

Teksto modelis	**Paveikslėlių modelis**	**MicroAccuracy**	**MacroAccuracy**
Averaged Perceptron	Inception V3	0.87	0.87
Decision Tree	Inception V3	0.814	0.829
Linear Support-Vector Machine	Inception V3	0.91	0.915
Logistic Regression (L-BFGS)	Inception V3	0.937	0.921
Averaged Perceptron	ResNet50	0.871	0.872
Decision Tree	ResNet50	0.815	0.833
Linear Support-Vector Machine	ResNet50	0.909	0.915
Logistic Regression (L-BFGS)	ResNet50	0.941	0.935
Averaged Perceptron	ResNet101	0.868	0.869
Decision Tree	ResNet101	0.811	0.827
Linear Support-Vector Machine	ResNet101	0.908	0.914
Logistic Regression (L-BFGS)	ResNet101	0.937	0.925
Averaged Perceptron	MobileNet V2	0.871	0.873
Decision Tree	MobileNet V2	0.815	0.833
Linear Support-Vector Machine	MobileNet V2	0.91	0.916
Logistic Regression (L-BFGS)	MobileNet V2	0.94	0.934

Išvados

1. Iš tekstu paremto įvertinimo modelių, „Random Forest“ ansambliavimo modelis grąžina tiksliausius rezultatus.
2. Gali būti, kad galima geriau pritaikyti paveikslėlių sudėtingumo įvertinimą, jei rastas labiau tokio tipo atpažinimui skirtas modelis.
3. Dvišakis neuroninis tinklas labai efektyviai veikia. Jį pritaikius, gaunamas žymiai didesnis tikslumas (80-90%), lyginant su vien tekstu (71%) ar paveikslėliais (43%) paremtais modeliais.
4. Yra galimybių daugiau patobulinti dvišakio neuroninio tinklo tikslumą, pvz. atliekant apjungimą prieš gaunant galutinius rezultatus iš dviejų buvusių modelių.
5. Dar atlikus papildomų pakeitimų, šiuo variantu būtų galima pasinaudoti kokybiškai įvertinti programavimo uždavinių sudėtingumą.
6. Automatinė uždavinių sunkumo įvertinimo sistema praeityje buvo sukurta, tačiau sukurtos versijos neprieinamos.
7. Pagal atliktą tyrimą galima sakyti, kad sistemai reikiami komponentai daugumoje sričių jau egzistuoja.
8. Rinkoje yra įvairūs įrankiai bei modeliai, atliekantys teksto arba paveikslėlių atvaizdavimą – jais naudojantis turėtų būti galima lengviau arba efektyviau atlikti darbą.
9. Rinkoje yra įvairūs įrankiai, skirti sukurti ir pritaikyti neuroninius tinklus.
10. Teisingai įvertinti sudėtingumą šio tipo programinei įrangai geriausia įvertinant ir paveikslėlius, ir tekstą, taip pat tikrinant visko poziciją – tikriausiai geriausia bandyti ir teksto, ir paveikslėlių analizę, taip pat juos abu apjungiant.

Literatūros sąrašas

1. „Tools of Evaluation in Education,“ 14 01 2020. [Tinkle]. Available: https://physicscatalyst.com/graduation/tools-of-evaluation-in-education/.
2. „Ugdymo plėtotės centras,“ [Tinkle]. Available: https://www.upc.smm.lt/naujienos/30_straipsnis/mokymai/geografija/UZDUOCIU_SUDARYMAS_IR_VERTINIMAS.ppt. [Kreiptasi 05 01 2019].
3. Q. L. E. C. H. Z. M. G. S. W. Y. S. G. H. Z. Huang, Question difficultyprediction for reading problems in standard tests, in: Proceedings of the AAAI Conferenceon Artificial Intelligence,, Z. Huang, 2017.
4. G. B. I. H. C. P. J. P. C. Koutsojannis, Using a hybrid AI approach for exercise difficulty level adaptation, International Journal ofContinuing Engineering Education and Life Long Learning 17, 2007.
5. „Summary Statistic Selectionwith Reinforcement Learning,“ [Tinkle]. Available: http://uu.diva-portal.org/smash/get/diva2:1342908/FULLTEXT01.pdf. [Kreiptasi 14 01 2020].
6. D. D. Palmer, Text preprocessing. Handbook of natural language processing 2, 2010.
7. J. J. Webster, „Tokenization as the initial phase in NLP,“ 1992.
8. T. Michel, R. Tesar ir J. Karel, „Influence of Word Normalization on Text Classification,“ 2006.
9. Y. ZHANG, R. JIN ir Z.-H. ZHOU, „Understanding bag-of-words model: a statistical framework,“ 2010.
10. J. Ramos, „Using TF-IDF to Determine Word Relevance in Document Queries,“ 2003.
11. R. N. F. Y. L. N. A. W. S. D. B. A. J. Myles, An introduction to decisiontree modeling, Journal of Chemometrics: A Journal of the Chemometrics Society 18, 2004.
12. M. R. Segal, „Machine learning benchmarks and random forest regression,“ 2004.
13. Q. M. T. F. T. W. W. C. W. M. Q. Y. T.-Y. L. G. Ke, LightGBM: A highlyefficient gradient boosting decision tree, Advances in neural information processing sys-tems 30, 2017.
14. J. N. D. M. H.-J. S. P. T. P. T. R. Bollapragada, A progressive batching L-BFGS method for machine learning, in: International Conference on Machine Learning, PMLR, 2018.
15. Y. Goldberg ir M. Elhadad, „Learning sparser perceptron models,“ 2011.
16. G. Fung, O. L. Mangasarian ir J. W. Shavlik, „Knowledge-Based Support Vector Machine Classifiers,“ 2002.
17. A. F. E. B. H. B. F. H. M. Galar, An overview of ensemblemethods for binary classifiers in multi-class problems: Experimental study on one-vs-oneand one-vs-all schemes, Pattern Recognition 44 (, 2011.
18. R. Szeliski, Computer vision: algorithms and applications, Springer Science & BusinessMedia, 2010.

19. R. N. K. O'Shea, An introduction to convolutional neural networks, arXiv preprintarXiv:1511.08458, 2015.
20. „An Introduction to Convolutional Neural Networks," [Tinkle]. Available: https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/. [Kreiptasi 28 12 2020].
21. T. Hothorn ir B. Lausen, „Double-bagging: combining classifiers by bootstrap aggregation.," 2003.
22. J. A. e. a. Hoeting, „Bayesian model averaging: a tutorial".
23. K. e. a. Monteith, „Turning Bayesian model averaging into Bayesian model combination," 2011.
24. M. M. e. a. Al Rahhal, „Learning a multi-branch neural network from multiple sources for knowledge adaptation in remote sensing imagery," 1890.
25. „About TensorFlow," [Tinkle]. Available: https://www.tensorflow.org/about. [Kreiptasi 21 12 2020].
26. „About Scikit," [Tinkle]. Available: https://scikit-learn.org/stable/about.html. [Kreiptasi 14 01 2020].
27. „Mobile Game Test Automation With Image Recognition," [Tinkle]. Available: http://jultika.oulu.fi/files/nbnfioulu-201802131231.pdf. [Kreiptasi 14 01 2020].
28. „About ML.NET," [Tinkle]. Available: https://dotnet.microsoft.com/apps/machinelearning-ai/ml-dotnet.
29. M. R. SEGAL, „Machine learning benchmarks and random forest regression," 2004.

Priedai

1 priedas. Straipsnis, parašytas ICIST 2021

Automatic programming problem difficultyevaluation – first results

Abstract. In this work, we address automatic evaluation of the difficulty of programming problems or exercises. Typically the problems consist of both text description and accompanying figures. We collect a suitable dataset, investigate the evaluation based on the text and theimage data separately, as well as a combination of the two. The first results of this investigation are reported, together with the discussion and future work.

Keywords: Programming problem, Difficulty evaluation, Natural language processing, Convolutional neural network, Deep learning, Machine learning.

1. Introduction

Programming problems of various kinds are regularly used to teach those interested in programming how to do so effectively in a practical way, and on occasion, they are also utilized to evaluate or demonstrate the ability of an individual regarding a particular field.

The issue with these sorts of problems, however, is that compared to other methods, these kinds of programming problems tend to be comparatively subjective - there is oftentimes very little objective way to evaluate exactly how difficult an exercise would be due to those writing them having a different perception of it compared to everyone else.

This project is an attempt to resolve part of this issue by utilizing machine learning to evaluate difficulty in regards to programming problems. Utilizing this system, it should be possible to have a more objective view of a problem and make it easier to perform the aforementioned teaching and evaluation of skills if successful.

The problem formulation usually includes both text and images, which makes this task more difficult, since typically different machine learning methods are used for the two types of data. Here we attempt to investigate the difficulty estimation based both on text, rendered image of the problem, and combination of both.We review the previous related work in Section 2, explain the data that we use in Section 3, present our methods used and preliminary results in Section 4, and finally, a short discussion is given, alongside ways to improve it in thefuture, in Section 5.

2. Related Work

Here we review the basic approaches and algorithms used for this type of problem. Due to the unique nature of the task, exact methods for evaluation have not been fully considered. As such, a wider variety of methods is considered.

2.1. Approaches

There have not been many attempts to approach this problem in particular, but there have been attempts to achieve results in similar fields. One example in particular is an attempt at predicting the difficulty of various questionsin reading problems [5] that utilized a particular framework called the Test-aware Attention-based Convolutional Neural Network (TACNN). According to the given results in the paper, the approach does improve on the result by some amount, and experts generally have lower accuracy on the evaluation than theresulting neural network.

Another notable approach has been the utilization of a hybrid AI [7] to evaluate exercise difficulty. To initialize the system, a teacher, as an expert, needs to input a rule set. After this is done, the feedback of any students that have taken the exercise is taken and used to adjust the results via a genetic algorithm. The results obtained from such indicates that a significant portion of exercises are initially evaluated incorrectly. However, this approach appears to be flawed - the method used indicates exactly one set of rules per difficulty level, which is fairly inaccurate to a realistic situation. In addition, if multiple rules for a single difficulty level are included in the starting rule set, the system cannot correctly determine which to use as the starting point to adjust from. According to the article, the rule to be used as a base for the genetic algorithm to adjust was chosen at random at the time of writing.

Beyond this, there do not appear to be many other methods utilized for tasks of this nature.

2.2. Algorithms

There are numerous algorithms that can be utilized. As the method we have been using (detailed in the Methods section) involves both text and images, there will be two subsections for algorithms associated with both, as well as another subsection indicating ways to combine multiple methods into one.

Text-Based Evaluation Natural Language Processing (NLP) initially requires some amount of pre-processing for the text. [10] There are multiple parts of pre-processing that can be utilized, as well as many different methods, so only a small portion will be mentioned here:

1. **Tokenization** is a process during which large amounts of text are separated and occasionally classified into smaller sections that machine learning can utilize effectively. Commonly used methods involve separating text by sentences or separating text by words.

2. **Normalization** is a method utilized to improve a system by attempting to remove redundant words with similar meanings. **Stemming**, for instance, is a method used to reduce words to their root form, and **lemmatization** removes prefixes and suffixes from words.

As an example, the **bag-of-words** method gathers all of the tokens obtained from a text into a "bag" – a format that completely ignores word order and grammar, but retains the number of times each word has shown up in a text in a way that a computer can use in the future.

For text-based evaluation, various algorithms can be used. To simplify things,\just some of the useful methods that have been evaluated will be described here.

1. **A decision tree** [8] is a simple method for classification. It makes use of multiple true/false statements to form a result. A decision tree classifier in machine learning forms this sort of tree through the use of training data to evaluate the exact values necessary. It is commonly used for relative simplicity and effectiveness, and there are several methods to improve the accuracy of these kinds of methods as well (detailed further under Model combination). A simple example of the way a decision tree works is shown in Figure 1 – In this case, Y is a binary variable, while the other two variables, X1 and X2, are used in this case to determine what the result should be.

   Once a model is formed, pruning can be performed for the sake of lowering the redundant or otherwise excessive branches and simplify the resulting model. While unnecessary, it can help with speed on several occasions.

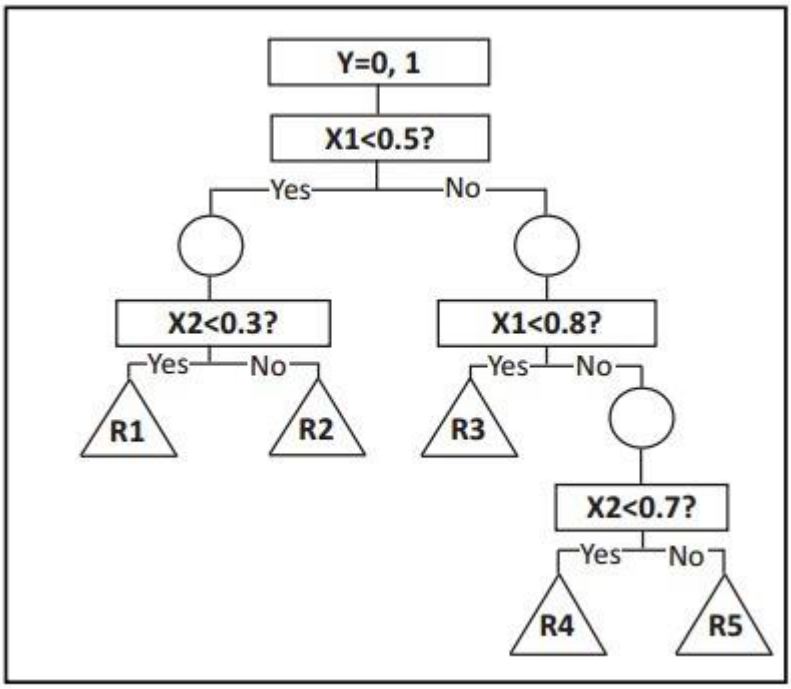


Figure 1. Sample decision tree based on binary target variable Y. The image is used from [8] under the Creative Commons Attribution-NonCommercial-Share Alike 4.0Unported License.

2. Light Gradient Boosting Machine (**LightGBM**) [6] is a gradient boost-ing decision tree (GBDT), designed by Microsoft. In particular, this imple-mentation adds a couple of techniques: Gradient-based One-Side Sampling (GOSS), which allows for an accurate information gain with less data, and Exclusive Feature Bundling (EFB), which bundles mutually exclusive features together in a way that would allow for fewer variables that the model has to look into. Through this, the resulting method is approximately 20 times faster while providing roughly equivalent accuracy.
3. The Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm (**L-BFGS**) [1], is a method to approximate the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm using a limited amount of available memory. Oftentimes this method is used for parameter estimation. The algorithm estimates an inverse Hessian matrix to help with determining the exact value.

Some classification methods are only created for the purposes of binary classification. To change these into multiclass classification, there are two methods to go about this, in particular [3]:

1. **One-vs-All** method creates n different versions of the same model, where n is the number of different classifications available, with each one having two different possible results - it is one of the available classifiers, or it is any of the others. When evaluating, the resulting model returns the classifier for which the model had the highest result. This method is generally faster and requires less memory to utilize.

2. **One-vs-One** method creates a different model for each possible combination of classifiers available, with each one having two of the possible multiple classifiers available. Once all of the different models finish, the correct answer is chosen by summing up or averaging all scores, and returning the highest result. This method is more complicated, slower, and requires more memory to utilize, but it may obtain more accuracy on occasion compared to One-vs-All.

Image-based Evaluation Computer vision is, to put it simply, an attempt to make a computer interpret vision in a similar way any other individual does [11]. Thus far, most cases of computer vision involve extremely specialized forms, such as optical character recognition (OCR) to interpret symbols from images. While more general-purpose methods do exist, they are not quite applicable to the task at hand presently, though there is a good chance that it will be more possible to use in the future.

For image-based evaluation, the main thing that is used is neural networks – a network made of several layers of interconnected artificial neurons that take in a value and return a result based on that value. The most common form of these neural networks that are utilized are convolutional neural networks [9]. Various sorts of variations and advancements have been made over the years, but as a general case, these sorts of neural networks rely on three forms of neuron layers:

1. **Fully-connected layers** are regularly used in all forms of artificial neural networks. The main characteristic of these layers is that they are completely connected to all adjacent layers, without being connected in any way to any other layers.
2. **Convolutional layers**, which are capable of learning various kernels that are then utilized to more efficiently evaluate a given image, are the primary layers the convolutional neural network utilizes. In most standard neural networks, reading an image would regularly result in a model that becomes too large to train in any effective manner. By using a convolutional layer, it is possible to reduce the complexity of a model significantly.
3. **Pooling layers**, which aim to pool together values in an attempt to gradually reduce the dimensions of the received image in such a way that it could be reliably used without making the system too complex.

Model Combination Several different methods have been used to put together multiple models in a way that would make the end result more capable than any of the individual parts. Oftentimes, these models are mostly identical to the others it is being combined with, excluding the data it was trained with. This sort of model combination is referred to as ensemble learning. The most common examples of this are [12]:

1. **Bootstrap aggregating**, sometimes referred to as **bagging**, is by far one of the simplest methods of ensemble learning available. For this method, the data is randomized during learning for each of the base learners. The results are counted for each of the available results, then these results are taken and averaged. The value that has achieved the highest average is considered to be the correct result. One example of this is random forests - an ensemble model that makes use of many different decision trees to form a result, regularly more accurate than any of the individual decision trees. It is a particularly quick to create model, but it is significantly less accurate compared to the others.
2. **Bayesian model averaging** [4] is created in a fairly similar manner to bootstrap aggregating – the data is randomized during learning, then an overarching model puts together the result. The key difference, however, is that each of the individual models in this exact scenario has a set weight during the evaluation, influencing the end result in this way. The accuracy is significantly better compared to Bootstrap aggregating due to this.

Aside from this, there is another method to utilize multiple different models, however this one requires more specific methods - **two-branch neural networks**. [2] These networks utilize two different models - for example, one that takes in text and another that takes in images, obtains the partial results, then runs those results through another layer of neurons to get a final result. By making use of this, both text and images are evaluated together.

3. Datasets

The dataset has been created personally through web scraping from several free websites that can be utilized, such as hackerrank.com and codechef.com.

The finalized dataset contains over 5000 different entries, each one containing difficulty, accuracy (a percentage of how many individuals successfully provided a suitable answer to the problem), the problem name, a description, and a variable number of images.

After this is finished, the data is recreated into multiple other methods to ensure simplicity for the actual evaluation. A CSV file is created from the text data (difficulty, problem name, description), and image data is combined with the description to create a set of images that contain the necessary text for evaluation.

4. Methods and Results

The methods we have utilized involve several different text classification methods to obtain different results – perceptrons, linear support vector machines, etc. A two-branch neural network is also used to improve the evaluation by adding deep learning-based image recognition. A specially-made program is used to obtain results and return the possible accuracy.

4.1. Text-based Evaluation

The results for text evaluation are fully detailed in Table 1. In this case, “MicroAccuracy” refers to an accuracy calculation from all available results, while “MacroAccuracy” takes the precision and recall in place of it instead.

The text preprocessing used is as follows: Each of the possible end result labels is assigned a numerical key, then all of the text from the other two possible inputs is transformed into a vector of floats representing counts of n-grams (an identical continuous sequence of n items) for both words and characters. Through doing this, it can then evaluate what each word indicates for a problem’s difficulty.

For the sake of a more comprehensible explanation, the basic way accuracy is calculated is

$$P_r = \frac{TP}{TP + FP}$$

where TP refers to the number of true positives, and FP refers to the number of false positives.

Micro-average Accuracy, written here as MicroAccuracy, refers to adding all of the values together before calculations. So, as an example, if there were three classes to evaluate from, micro-average accuracy would be counted as

$$P_{r_{micro}} = \frac{TP1 + TP2 + TP3}{TP1 + TP2 + TP3 + FP1 + FP2 + FP3}$$

Macro-average accuracy, written here as MacroAccuracy, refers to dividing each of the results before averaging them, thus ensuring that each class has an equal contribution to the result. Using the same example of three classes, this is how macro-average accuracy is calculated as.

$$P_{r_{macro}} = \frac{TP1}{TP1 + FP1} + \frac{TP2}{TP2 + FP2} + \frac{TP3}{TP3 + FP3}$$

Micro-average accuracy is preferable when trying to calculate accuracy for a multi-class classification problem if it is believed that there may be some imbalance between the number of entries per class. In this case, macro-average accuracy is used to show that, while there is a small amount of imbalance in the data, it does not truly impact the result with most models.

Table 1. Text evaluation results

Model name	**MicroAccuracy**	**MacroAccuracy**
Random Forest	0.718	0.725
Decision Tree	0.711	0.712
LightGBM	0.701	0.708
Logistic Regression (L-BFGS)	0.660	0.657
Maximum Entropy (SDCA)	0.639	0.661
Stochastic Gradient Descent	0.620	0.623
Averaged Perceptron	0.609	0.641
Linear Support-Vector Machine	0.583	0.597
SymbolicSgdLogisticRegression	0.545	0.584
Maximum Entropy (L-BFGS)	0.458	0.319

4.2. Image-Based Evaluation

For image-based evaluation, a single image containing the whole problem description together with the illustrations was rendered for each problem. The size needed for the image is pre-calculated, the text and images are taken, then the text is rendered using a predefined style, with images being inserted in places where they were in the original problem. An example of the image formed from the text and images of an exercise is shown in Figure 2. Once that is finished, the result is used as data for a deep neural network, by which the data is evaluated. Several well known models were used – their names as well as the obtained results are shown in Table 2.


Ada School

Read problems statements in Hindi, Mandarin Chinese, Russian, Vietnamese and Bengali as well.
Ada's classroom contains
N * M
tables distributed in a grid with
N
rows and
M
columns. Each table is occupied by exactly one student.
Before starting the class, the teacher decided to shuffle the students a bit. After the shuffling, each table should be occupied by exactly one student again. In addition, each student should occupy a table that is adjacent to that student's original table, i.e. immediately to the left, right, top or bottom of that table.
Is it possible for the students to shuffle while satisfying all conditions of the teacher?
Input
The first line of the input contains a single integer
T
denoting the number of test cases. The description of
T
test cases follows.
The first and only line of each test case contains two space-separated integers
N
and
M
.
Output
For each test case, print a single line containing the string "YES" if it is possible to satisfy the conditions of the teacher or "NO" otherwise (without quotes).
Constraints
1≤T≤5,000
2≤N,M≤50
Example Input
2
3 3
4 4
Example Output
NO
YES
Explanation
Example case 2: The arrows in the following image depict how the students moved.
All submissions for this problem are available.
Author: 7★alei
Editorial: https://discuss.codechef.com/problems/ADASCOOL
Tags: alei, conditional-statement, matrices, simple
Date Added: 31-08-2018
Time Limit: 1 secs

Figure 2. Example of an image formed from a problem's text and images. The problem used is from codechef.com.

Table 2. Image evaluation results

Model name	**MicroAccuracy**	**MacroAccuracy**
ResNet50	0.403	0.343
ResNet101	0.436	0.385
MobileNet V2	0.297	0.239
Inception V3	0.339	0.258

4.3. Combining Text and Image-Based Evaluation

In an attempt to improve the accuracy of the text-based and image-based methods, a two-branch neural network was used. The two separate models are used initially – one for text, the other for images. The outputs of the two models are then concatenated and re-evaluated to obtain the refined result. The two-branch neural network we are using is presented in Figure 3. For now, we use a single layer neural network for the results refinement part.

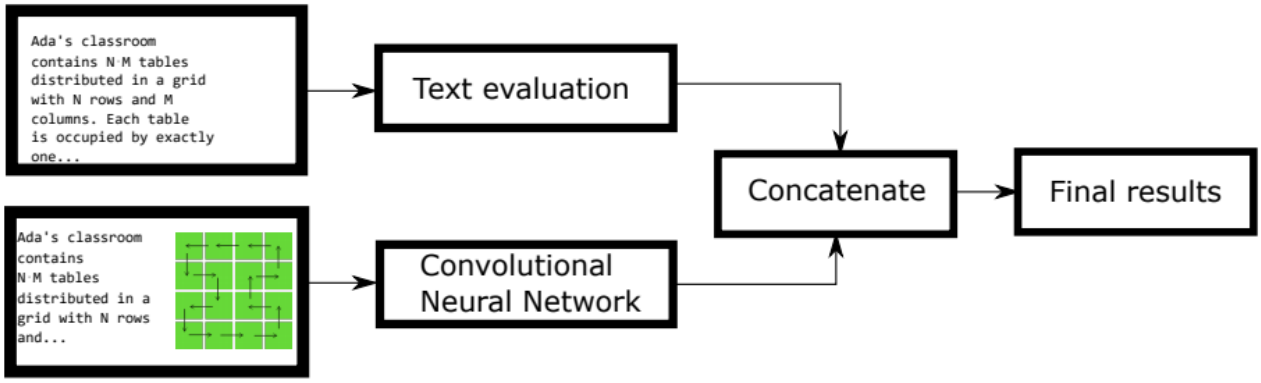


Figure 3. Example of a two-branch neural network being used.

Before using the two-branch neural network, our image classification accuracy was about 40-50 %. When a two-branch neural network is used, the accuracy goes up rapidly, up to about 80+ % – significantly higher than either of the individual models.

Text model name	Image model name	MicroAccuracy	MacroAccuracy
Averaged Perceptron	ResNet50	0.871	0.872
Decision Tree	ResNet50	0.815	0.833
Linear Support-Vector Machine	ResNet50	0.909	0.915
Logistic Regression (L-BFGS)	ResNet50	0.941	0.935

5. Discussion and Future Work

Among text-based models, with a bag of n-grams preprocessing, the Random Forest ensemble model returned the best results.

Future work for text-based models may include the use of different text preprocessing methods – the completely different structure of the input may increase the accuracy of the models further.

More options for the image recognition network could be investigated in the future.

The two-branch neural network is particularly useful at the moment. The accuracy of the two-branch neural network is significantly greater (80 %) than either the text-based (72 %) or image-based (43 %) models on their own.

More options for the combination of the two branches could be considered in the future, including a single fully trainable architecture with no intermediate interpretable results from the text and image-based models.

References

1. Bollapragada, R., Nocedal, J., Mudigere, D., Shi, H.J., Tang, P.T.P.: A progressive batching L-BFGS method for machine learning. In: International Conference on Machine Learning. pp. 620–629. PMLR (2018)
2. Chen, H., Lagadec, B., Bremond, F.: Partition and reunion: A two-branch neuralnetwork for vehicle re-identification. In: CVPR Workshops. pp. 184–192 (2019)
3. Galar, M., Fern´andez, A., Barrenechea, E., Bustince, H., Herrera, F.: An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes. Pattern Recognition 44(8), 1761–1776 (2011)
4. Hoeting, J.A., Madigan, D., Raftery, A.E., Volinsky, C.T.: Bayesian model averaging: a tutorial. Statistical science pp. 382–401 (1999)
5. Huang, Z., Liu, Q., Chen, E., Zhao, H., Gao, M., Wei, S., Su, Y., Hu, G.: Question difficulty prediction for reading problems in standard tests. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 31 (2017)
6. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y.: LightGBM: A highly efficient gradient boosting decision tree. Advances in neural information processing systems 30, 3146–3154 (2017)
7. Koutsojannis, C., Beligiannis, G., Hatzilygeroudis, I., Papavlasopoulos, C., Prentzas, J.: Using a hybrid AI approach for exercise difficulty level adaptation. International Journal of Continuing Engineering Education and Life Long Learning 17(4-5), 256–272 (2007)
8. Myles, A.J., Feudale, R.N., Liu, Y., Woody, N.A., Brown, S.D.: An introduction to decision tree modeling. Journal of Chemometrics: A Journal of the Chemometrics Society 18(6), 275–285 (2004)
9. O'Shea, K., Nash, R.: An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458 (2015)
10. Palmer, D.D.: Text preprocessing. Handbook of natural language processing 2, 9–30 (2010)
11. Szeliski, R.: Computer vision: algorithms and applications. Springer Science & Business Media (2010)
12. Zhou, Z.H.: Ensemble learning. Encyclopedia of biometrics 1, 270–273 (2009)