UNIVERSITETI I PRISHTINËS

FAKULTETI I INXHINIERISË ELEKTRIKE DHE KOMPJUTERIKE

PUNIM DIPLOME

Tema:

Detektimi dhe reduktimi i spam emailave me ML dhe hash algoritmet

Mentori:
Prof. Dr. Blerim Rexha

Kandidati:
Lumi Bytyçi

Prishtinë, Prill 2020

Abstrakt

Që nga zbulimi i postës elektronike në vitin 1972 kjo formë e komunikimit ka shënuar rritje dhe sot është njëra prej formave më dominante të komunikimit (miliarda email dërgohen dhe pranohen për çdo ditë). Në formimin e email-it si medium komunikimi është investuar shumë në krijimin e infrastrukturës përkatëse teknologjike dhe protokolleve mbështetëse të komunikimit, dhe si rrjedhojë email ka arritur në nivel të lartë të stabilitetit dhe maturisë. Sot, email përdoret për komunikim formal e joformal, për komunikim në ambiente shkollore dhe biznesore. Në ngritjen e email ndikuan sidomos lehtësia dhe kostoja e ulët e dërgimit. Por, si çdo formë tjetër e komunikimit, edhe email është subjekt i keqpërdorimit. Format më mbizotëruese të keqpërdorimit janë dërgimi i spam, spoofing dhe phishing, që të gjitha keqdashëse për marrësin. Tema qendrore e këtij punimi është dukuria e dërgimit të spam email, apo më saktësisht, krijimi i mekanizmave mbrojtës për detektim dhe reduktim të spam email. Në shërbim të nevojës që të trajtohen detajisht dy aspektet e mekanizmave mbrojtës, pra detektimit dhe reduktimit, këto tema do të trajtohen veçanërisht. Për detektimin e spam email dhe rrjedhimisht filtrimin e saj ekzistojnë shumë algoritme dhe metoda të ndryshme që sot zbatohen nga ofruesit e shërbimeve të email. Në këtë punim trajtohet metoda e detektimit duke shfrytëzuar arritjet në fushën e Machine Learning si nënfushë e inteligjencës artificiale (AI). Do të shfrytëzohen rrjetat neurale artificiale të tipit LSTM në kombinim me disa metoda të procesimit të gjuhës natyrale (natural language processing) për krijimin e një modeli të përshtatshëm për klasifikim binarë të email në spam ose jo spam. Për t'iu shmangur procesit të mbledhjes, pastrimit (procesimit) dhe etiketimit të email që nevojiten për procesin e trajnimit të modelit është shfrytëzuar një dataset me mijëra email të etiketuara dhe të gatshme për trajnim. Në anën tjetër, në funksion të reduktimit të dërgimit të spam email do të trajtohet shfrytëzimi i konceptit të vërtetësisë së punës (proof of work) koncept ky shumë i njohur sidomos në botën e blockchain që shfrytëzon gjerësisht hash algoritmet. Në fund janë dhënë edhe rekomandimet përkatëse për mbrojtje nga spam email dhe janë cekur edhe mundësitë e zgjerimit të këtij studimi në të ardhmen.

Abstract

Ever since its invention in 1972, the electronic email (or email for short) has always been a growing trend. Today, email is one of the most dominant forms of communication with billions of emails being sent and received daily. A lot has been invested in shaping email as a medium by creating the necessary communication protocols and technological infrastructure. Consequently, email has achieved a high stability and maturity status as a communication medium. Email is used for formal and informal communication and also for education and enterprise communication. Most influential factors that dictated the growth of emails are the ease of use and low cost of communication. As with all technologies, email is also subject to misuse. Most prevalent types of misuse are spam, spoofing and phishing, which are all unsolicited for the receiver. The phenomenon of spam email and the creation of mechanisms to detect and reduce this phenomenon are the central subject of this thesis. For the sake of explaining both of these aspects, namely, detection and reduction, they will be analyzed separately. There are numerous algorithms and methods in existence that accomplish the detection of spam emails. This thesis treats methods of spam email detection that make use of Machine Learning as a sub-field of Artificial Intelligence (AI). The methods that will be explored utilize LSTM neural networks and several concepts from the field of Natural Language Processing (NLP) to help create a binary classifier that classifies emails into spam and not spam. To avoid the process of collecting, processing and labeling of emails for the phase of model training, a ready-made dataset with thousands of labeled emails is used. In function of spam email reduction, a mechanism that utilizes the concept of *proof of work* is used. This concept is very well known in the world of cryptography and blockchain and makes extensive use of hash algorithms. At the end, relevant recommendations for protection against spam email are given, and also the possibilities of expanding on this study in the future.

Falënderime

Falënderoj familjen time, veçanërisht prindërit, për mbështetjen e pakushtëzuar në çdo aspekt të rrugëtimit tim. Falënderoj të gjithë profesorët për përkushtimin dhe motivimin e dhënë gjatë studimeve. Falënderoj veçanërisht profesorin Dr. Blerim Rexha për këshillat dhe ndihmën gjatë realizimit të punimit. Gjithashtu falënderoj të gjithë kolegët dhe miqtë e mi që më qëndruan pranë dhe më mbështetën gjatë studimeve.

Përmbajtja

Tabela e Figurave

1 Hyrja

1.1 Hyrje

Komunikimi përmes postës elektronike (email) është njëra prej formave më dominante të komunikimit në botën digjitale në të cilën jetojmë. Sipas statistikave, vetëm në vitin 2019 janë identifikuar 3.9 miliard përdorues të email anembanë botës, dhe parashikohet që ky numër të mbërrijë shifrën 4.3 miliard në vitin 2023, siç është paraqitur tek Fig. 1 [1].

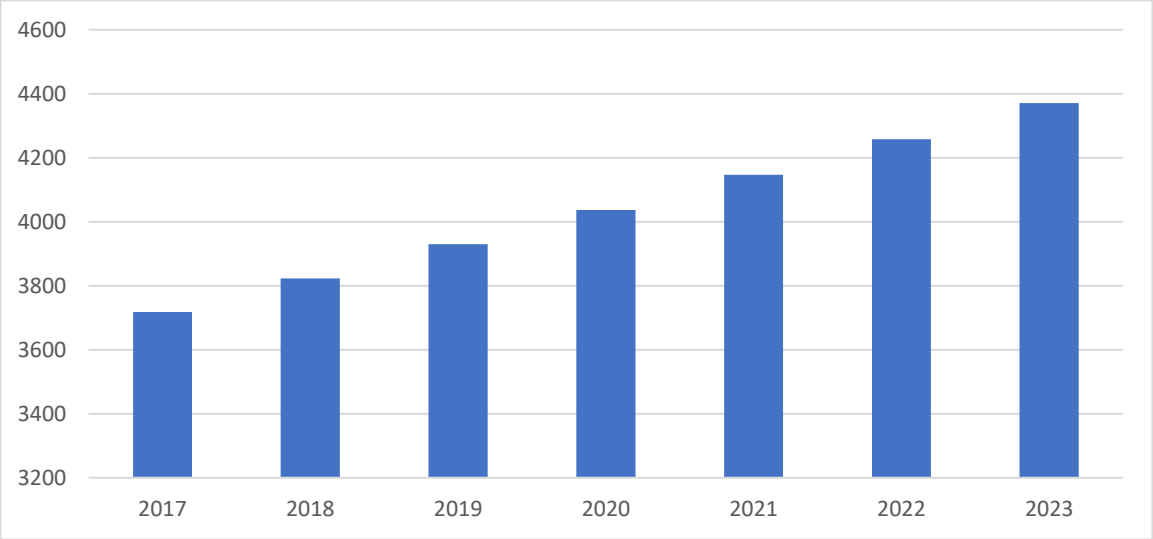

Fig. 1: Numri i përdoruesve të email në botë nga 2017 *deri në vitin 2023 (në miliona) [1]*

Studimet tregojnë se për çdo ditë në vitin 2019 janë dërguar dhe pranuar rreth 293.6 miliard email. Kjo konvertohet në 3.3 milion email për çdo sekondë [2]. Sikurse edhe çdo formë tjetër e komunikimit, edhe komunikimi përmes email është subjekt i keqpërdorimeve. Forma më mbizotëruese e keqpërdorimit janë spam email. Spam apo junk konsiderohen email-at e padëshiruara të dërguara me shumicë tek një liste e marrësve, zakonisht për qëllime komerciale [3]. Spam email i parë është dërguar në maj të vitit 1978. Përmbajta e atij email kishte qëllim promovimin e një modeli të ri të kompjuterit dhe iu dërgua 393 personave [4]. Kjo dukuri sot ka përmasa enorme dhe është trend në rritje. Sipas statistikës, vetëm në vitin 2014 është vlerësuar se 90% e email-ave të dërguara janë klasifikuar si spam [5]. Spam apo junk email nuk kanë për qëllim vetëm marketingun, por edhe shpërndarjen e faqeve të pa sigurta, vjedhjen e informatave dhe mashtrimin e përdoruesve. Si e këtillë, viteve të fundit, dërgimi i spam email është shndërruar nga një dukuri bezdisëse, në një dukuri joligjore dhe profitabile. Si rrjedhojë, ka lindur nevoja që të krijohen mekanizma që mbrojnë përdoruesit nga këto lloje të emailave. Njëherazi, krijimi i mekanizmave mbrojtës është edhe tema kryesore që trajtohet në këtë punim. Punimi ka për qëllim përzgjedhjen dhe vlerësimin e mekanizmave që mundësojnë detektimin e spam email me saktësi të lartë dhe reduktimin e tyre, gjithmonë duke pasur parasysh se detektimi shkon në funksion të reduktimit. Edhe pse sot ekzistojnë forma dhe algoritme të ndryshme që kryejnë funksionin e detektimit të spam email, ky punim fokusohet në zbatimin e Machine Learning apo më saktësisht shfrytëzimin e rrjetave neurale artificiale (Artificial Neural Networks) për klasifikim. Në rastin e studimit do të bëhet shfrytëzimi i teknikave moderne të procesimit të gjuhës natyrale (Natural

Language Processing) dhe të Machine Learning për trajnimin e një modeli që do të mundësoj klasifikimin e emailave në spam ose jo-spam, me saktësi të lartë.

1.2 Motivimi

Dërgimi i spam emailave është një dukuri negative me përmasa të jashtëzakonshme. Kur rezulton se shumica e emailave që dërgohen kategorizohen si spam, shumë lehtë mund të formulohet definicioni kontrovers se email është metodë komunikimi që si funksion primar ka shpërndarjen e spam, kurse dërgimi i informacionit të dobishëm është si funksion sekondar. Sigurisht se ky definicion nuk qëndron dhe e gjithë dukuria si negative duhet të luftohet dhe të reduktohet. Në fushën e sigurisë së Internetit, detektimi dhe reduktimi i spam emailave është temë me trajtim dhe rëndësi të veçantë. Janë krijuar mekanizma, skema dhe algoritme të lloj-llojshme që mundësojnë klasifikimin e emailave në spam ose jo spam. Sigurisht, disa metoda janë më të sakta e efektive se tjerat. Për këtë arsye, përmasat e problemit dhe metodat e shumta të "zgjidhjes" së problemit kanë shërbyer si nxitës kryesorë për trajtimin e kësaj teme në këtë punim.

1.3 Përshkrimi i problemit

Duke parë raportin shqetësues të emailave spam me ato jo-spam në totalin e emailave të dërguara dhe pranuara, është evidente se kërkohet nevoja për aplikimin e metodave dhe algoritmeve për zgjidhjen e këtij problemi. Në esencë, problemi është tek dërgimi i spam emailave ndërsa zgjidhja qëndron në detektimin e tyre. Është pothuajse e pamundur ndalimi i plotë i dërgimit të spam emailave, por është shumë e mundshme që ato të detektohen dhe filtrohen në anën e pranuesit në mënyrë që dëmi që shkaktojnë të reduktohet sa më shumë. Metoda e zgjedhur për detektimin e spam emailave në këtë punim është përmes shfrytëzimit të rrjetave neurale artificiale (artificial neural networks) si pjesë e lëmisë se inteligjencës artificiale. Hapi i parë që duhet të ndërmerret në këtë rast është sigurimi i setit me të dhëna (dataset) i cili do të përdoret për mësimin apo trajnimin e modelit. Dataseti nuk është asgjë më shumë se një grumbull me të dhëna, në këtë rast, një grumbull i emailave të dërguara me përmbajtje të ndryshme spam ose jospam. Është ndihmesë shumë e madhe nëse paraprakisht dataseti është i përpunuar dhe i etiketuar[1] (labeled). Një pjesë e konsiderueshme e datasetit ndahet për qëllime të testimit dhe të vlerësimit të performancës së modelit pas procesit të trajnimit. Hapi tjetër dhe mbase kryesor është përcaktimi i strukturës së modelit që do të përdoret. Duhet zgjedhur lloji i rrjetës artificiale neurale, gjithashtu edhe lloji dhe numri i shtresave. Pas këtij procesi mund të fillohet me procesin e mësimi apo trajnimit i cili mundëson kalibrimin e peshave (weights) dhe parametrave tjerë të secilës njësi të rrjetës artificiale neurale. Ky hap rezulton në një model të gatshëm për klasifikimin e spam emailave. Si përfundim, do të testohet dhe vlerësohet saktësia e modelit në bazë të performancës dhe disa parametrave tjerë. Në shërbim të reduktimit të spam emailave do të shfrytëzohet koncepti i vërtetimit të punës (proof of work), koncept ky shumë i njohur në botën e blockchain dhe kriptografisë.

[1] Etiketimi i të dhënave në rastin e datasetit me emaila nënkupton se për secilën email ekziston etiketa përkatëse që tregon se a është ai email spam ose jospam.

2 Posta elektronike (E-mail)

2.1 Email si medium i komunikimit

Me hovin e madh të zhvillimeve në fushën e teknologjisë kompjuterike u paraqit nevoja për një formë më të shpejtë dhe fleksibile të komunikimit. Metodat ekzistuese asokohe për komunikim sikurse dërgimi i letrave përmes postës, thirrjet telefonike dhe dërgimi i fax-it nuk ofronin dinamikën e kërkuar edhe pse këto forma sot ende janë në përdorim. Si rrjedhojë, filluan punimet për krijimin e një mediumi të ri të komunikimit i cili do të shfrytëzonte teknologjinë e rrjetave kompjuterike. Kështu, në fillimin e viteve të 70-ta u krijua versioni i parë i postës elektronike (email) për përdorim brenda rrjetës të quajtur ARPANET (Advanced Research Projects Agency). ARPANET ishte një rrjetë kompjuterike eksperimentale e financuar nga departamenti i mbrojtjes të SHBA-ve. Sot, email ka marrë përmasa globale dhe shërben si medium komunikimi që realizohet përmes rrjetës më të madhe kompjuterike që të gjithë e njohim si Internet. Sikurse posta klasike, edhe posta elektronike është formë asinkrone e komunikimit, që nënkupton se përdoruesit mund të dërgojnë dhe lexojnë mesazhet kur t'iu përshtatet atyre, pa pasur nevojë për koordinim ndërmjet palëve [6]. Email si teknologji ka evoluar shumë nga versionet fillestare dhe sot përpos përmbajtjes tekstuale, mundëson dërgimin e dokumentave si *attachments* dhe vendosjen e imazheve.

2.2 Infrastruktura dhe protokollet

Duke marrë parasysh shtrirjen e gjerë dhe qëndrueshmërinë që ka email si teknologji, është me rëndësi që të shpjegohet infrastruktura që e mundëson dërgimin dhe pranimin e email. Komponentet kryesore që e përbëjnë email janë *mail user agents*, *mail servers* dhe protokolli SMTP (Simple Mail Transfer Protocol) [6]. Këto komponente dhe lidhja në mes tyre ilustrohet sikurse në Fig. 2:

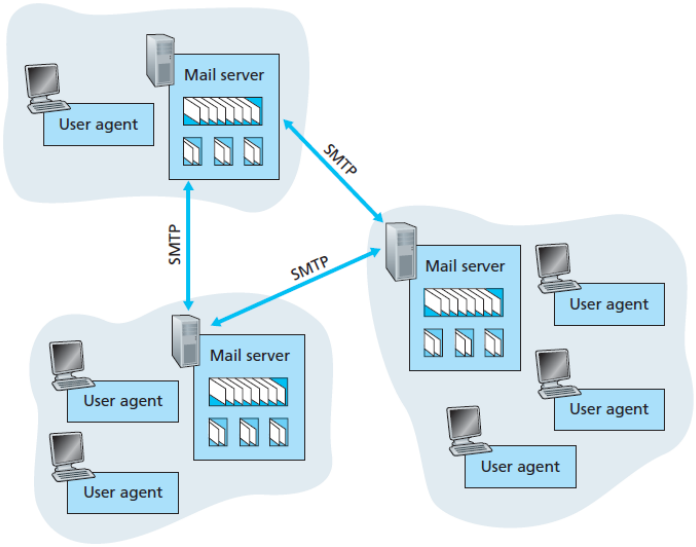

Fig. 2: Infrastruktura dhe protokollet e postës elektronike [6]

Mail user agent ose ndryshe i njohur si *email client* qëndron në skaj të kësaj infrastrukture dhe paraqet një program që mundëson leximin, shkrimin, ruajtjen dhe dërgimin e emailave. Këto programe mund të jenë të implementuara si web aplikacione siç është Gmail ose si desktop apo

mobile applications siç janë Microsoft Outlook, Apple Mail etj. Në këtë komponentë përdoruesit i ofrohet liria e zgjedhjes dhe ai mund të vendosë se cili shërbim i përshtatet më së shumti. Secili përdorues ka një kuti postare virtuale në server që në këtë kontekst quhet *mail server*. Mail serverët konsiderohen si pjesa kryesore e arkitekturës së email. Këta server do të pranojnë mesazhe nga user agent i përdoruesit dhe do t'i vendosin në një radhë të mesazheve. Mail serveri do të provoj të dërgojë këto mesazhe tek mail serveri përkatës i pranuesit. Pasi email mbërrinë tek mail serveri i marrësit, marrësi mund ta shkarkojë atë në user agent-in e vet. Për shkëmbimin e emailave në mes të mail serverit të marrësit dhe atij të dërguest përdoret protokolli i njohur SMTP. Në rrethana normale, komunikimi i mail serverëve nëpërmjet SMTP bëhet në mënyrë direkte (pa përdorur mail serverë ndërmjetës) edhe kur dy mail serverë gjenden në anë të ndryshme të globit [6]. Në fazat e para të adaptimit të email ishte normale që përdoruesit të kyçeshin vetë në mail serverin e tyre për leximin dhe shkrimin e emailave. Më vonë u krijuan protokolle që mundësojnë shkarkimin, dhe menaxhimin e emailave në mail serverin e pranuesit. Dy nga këto protokolle janë *POP3* dhe *IMAP*. Vlen të ceket se këto protokolle zbatohen në mes të mail serverit dhe user agent në anën e pranuesit, ndërsa prej dërguesit deri tek mail serveri i tij përdoret SMTP. Mail user agents që si bazë kanë web aplikacionet (sikurse Gmail) përdorin protokollin HTTP për komunikim me mail serverët.

2.3 Dukuria e spam email

Njëri prej definicioneve më të shkurta të spam thotë se spam konsiderohen emailat e padëshiruara. Spam emailat janë email që dërgohen në shumicë dhe kanë kryesisht për qëllim marketingun. Dekadave të fundit spam emailat kanë kaluar cakun e marketingut dhe kanë filluar të përdoren për të mashtruar përdoruesit e email. Format më të rrezikshme të spam kanë për qëllim mashtrimin, vjedhjen e informatave dhe infektimin me virus të kompjuterëve të shfrytëzuesve. Spam emailat që kanë për qëllim dëmtimin e përdoruesve i takojnë një kategorie tjetër të quajtur *phishing*. Për këtë arsye është investuar shumë në krijimin e sistemeve dhe mekanizmave mbrojtës për luftimin e kësaj dukurie. Një ndër metodat më efektive është tentimi i detektimit dhe filtrimit të këtyre lloje të emailave pasi ato të kenë mbërritur tek marrësit. Klasifikimi i emailave me këta mekanizma bëhet në dy kategori, spam dhe jo-spam ose ndryshe kategori e njohur si "ham" në terminologjinë e fushës së sigurisë në Internet. Statistikat tregojnë se çdo sekond dërgohen me miliona spam email [2]. Dërgimin e kësaj sasie kaq voluminoze e mundësojnë rrjetat e quajtura *botnet*. Botnet janë rrjeta të përbëra nga mijëra kompjuter privat të infektuar me viruse dhe që mund të kontrollohen nga pronari i rrjetës. Nga këto rrjeta dërguesit e spam email mund të nisin mijëra e mijëra spam email çdo sekond, dhe në të njejtën kohë ta mbajnë të fshehtë identitetin e tyre. Një shembull konkret i një botneti me përmasa gjigante është botneti i quajtur "Necrus". Një prej detyrave të këtij botneti ishte edhe dërgimi i spam emailave. Supozohej, se para se të luftohej dhe ndërprehej (deri në një masë) nga kompania Microsoft në mars të vitit 2020 [7], ky botnet zotëronte rreth nëntë milion kompjuter të infektuar në rrjet. Microsoft publikon se nga një vëzhgim 58 ditor, zbuluan se njëri nga kompjuterët, pjesë e këtij botneti, dërgoi një total prej 3.8 milion spam email tek më shumë se 40.6 milion përdorues [7]. Nuk ka nevojë për fakte shtesë për të konstatuar se sa e rrezikshme dhe e dëmshme është kjo dukuri. Përgjatë punimit shpjegohen disa nga metodat dhe mekanizmat që përdoren për luftimin e kësaj dukurie.

3 Machine Learning dhe hash algoritmet

3.1 Hyrje në Machine Learning

Machine Learning është shkenca dhe arti i programimit të kompjuterëve në mënyrë që ata të mund të mësojnë nga të dhënat [8]. Një definicion pak më gjeneral e ka dhënë Arthur Samuel në vitin 1959 kur thotë se Machine Learning (ML) është fushë studimi e cila merret me aftësimin e kompjuterëve që të mësojnë pa u programuar në mënyrë eksplicite. Është po i njejti studiues i shkencave kompjuterike i cili e popullarizoi termin Machine Learning i cili vetë është pjesë e fushës së inteligjencës artificiale. Edhe pse si fushë është mjaft e vjetër, njëra prej arritjeve që bëri bujë ishte identifikimi i shifrave të shkruara (handwritten digit recognition). Ky funksionalitet u krijua për herë të parë nga AT&T Bell Laboratories në vitin 1998 dhe u shfrytëzua nga shërbimi postar i SHBA për detektimin e kodeve postare [9]. Machine Learning sot gjenë zbatim në një numër jashtëzakonisht të madh të fushave të ndryshme. Disa zbatime të këtij koncepti që ka marrë hov të madh janë autopilotimi i veturave (self-driving cars), njohja e zërit dhe imazheve (image and voice recognition), bërja e parashikimeve, përmirësimi i rezultateve të makinave të kërkimit etj. Machine Learning kryesisht aplikohet në zgjidhjen e problemeve për të cilat nuk ekzistojnë zgjidhje tradicionale. Në shumicën e rasteve zgjidhja përmes aplikimit të algoritmeve të Machine Learning rezulton në kod me qartësi dhe performancë më të lartë [8]. Ideja gjenerale e shumicës së algoritmeve të Machine Learning është që të ndërtohet një model matematikor nga një mostër e të dhënave e pastaj të shfrytëzohet ky model për bërjen e parashikimeve [10].

3.2 Kategorizimi i sistemeve ML

Edhe pse u shpjegua ideja apo koncepti themelorë që përdoret në algoritmet e Machine Learning, ekzistojnë ndryshime apo nuanca në mënyrën se si mëson apo ushqehet modeli me të dhënat e trajnimit (training data). Duke marrë parasysh këto veçori, sistemet që përdorin Machine Learning mund të kategorizohen në bazë të teknikave në vijim:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Në vijim do të sqarohet shkurt kuptimi i secilës kategori të përmendur më lartë.

3.2.1 Supervised learning

Tek sistemet e Machine Learning që i takojnë kategorisë të mësimit të mbikqyrur (supervised learning), trajnimi i modelit bëhet nga një dataset i cili përmbanë edhe etiketat (labels) për secilën mostër. Për këtë arsye mund të thuhet se këto lloje të sistemeve mësojnë në mënyrë iterative[2] nga një numër i shumtë i mostrave (shembujve) dhe pastaj arrijnë në gjendje të kryejnë parashikime kur atyre i'u paraqitet një mostër e re. Një shembull specifik që i takon kësaj kategorie është pikërisht problemi i klasifikimit të email në spam ose jospam.

[2] Mostrat e datasetit për trajnim përdoren apo i ushqehen modelit më shumë se një herë

Sistemet e kësaj kategorie kryesisht shfrytëzohen për kryerjen e njërës nga funksionet në vijim:

- Klasifikim
- Regresion

Të dy këto funksione janë ilustruar grafikisht në Fig. 3**:**

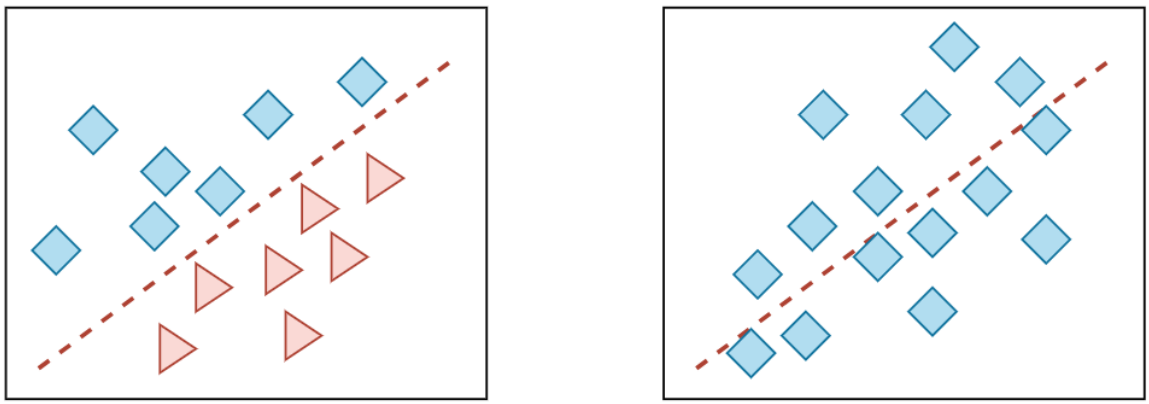

Fig. 3: Klasifikimi dhe regresioni të paraqitura grafikisht

Klasifikimi ka të bëjë me kategorizimin e të dhënave nëpër klasa apo kategori. Në bazë të numrit të klasave që ekzistojnë për kategorizim, kemi ndarjen si në vijim:

- Klasifikim binar (Klasifikim në dy klasa)
- Klasifikim në shumë klasa

Në anën tjetër, regresioni ka ngjashmëri shumë të madhe me klasifikimin, me dallimin e vetëm se tek regresioni bëhet parashikimi i një vlere të caktuar numerike. Një shembull ilustrues i regresionit do të ishte parashikimi i çmimit të një apartamenti duke u bazuar në lokacionin, vjetërsinë, hapësirën e shfrytëzueshme, numrin e dhomave etj.

Algoritmet më të përdorura që i takojnë kategorisë Supervised Learning janë:

- Support Vector Machines (SVM)
- Naive Bayes
- Linear Regression
- Decision Trees
- K-Nearest neighbor
- Neural Networks[3]

[3] Disa arkitektura të rrjetave neurale (neural networks) mund të konstrukohen ashtu që të jenë të pambikqyrura (unsupervised), shembuj janë autoenkoderët [6].

3.2.2 Unsupervised learning

Tek sistemet e Machine Learning që i takojnë kategorisë së mësimit të pambikqyrur (unsupervised learning) modeli do të mësojë apo trajnohet nga të dhëna për të cilat nuk ekziston etiketimi (labels). Nëse do merrej analogjia e mësimit klasik, atëherë kjo nënkupton se modelet e kësaj kategorie duhet të mësojnë vetë, pa ndihmën e një mësuesi, pra pa mbikqyrje.

Algoritmet që i takojnë kësaj kategorie janë shumë të përshtatshme për:

- Grupim (Clustering)
- Vizualizim dhe reduktim të dimensionalitetit
- Detektim të anomalive apo defekteve

Grupimi, siç është paraqitur tek Fig. 4 është një detyrë klasike e algoritmeve të kësaj kategorie. Një shembull konkret do të ishte grupimi i personave në bazë të shijeve të përbashkëta të filmave që shikojnë. Një model i pambikqyrur do të pranonte të dhëna (*features*) të pa etiketuar dhe do të mundohej të krijonte grupe në bazë të shijes së shikuesve.

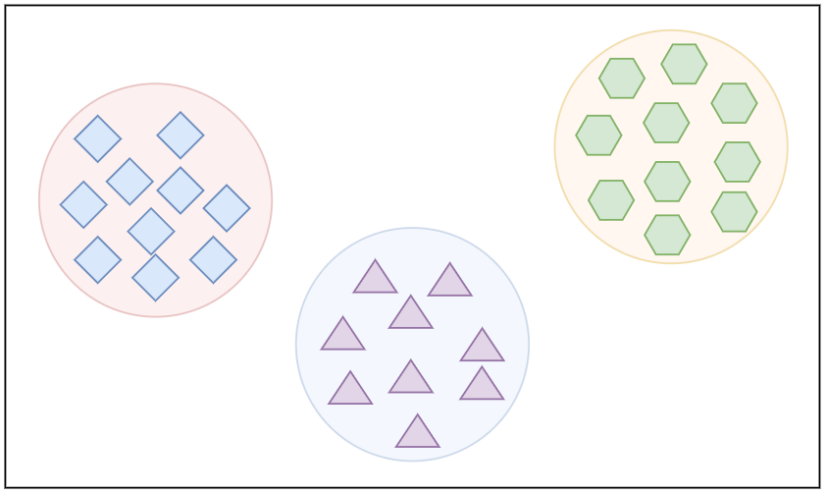

Fig. 4: Paraqitje grafike e grupimit (clustering)

Në kuadër të detektimit të anomalive, sistemet ML të kësaj kategorie kanë gjetur zbatim shumë të madh në detektimin e aktiviteteve ilegale siç janë mashtrimet me kartela të kreditit dhe transaksionet bankare të dyshimta [11].

Algoritmet më të shpeshta që i takojnë kategorisë Unsupervised Learning janë [8]:

- Grupim
  - K-Means
  - DBSCAN (Density-based spatial clustering of applications with noise)
  - Hierarchial Cluster Analysis

- Vizualizmi dhe reduktim i dimensionalitetit
  - Principal Component Analysis (PCA)

  - Kernel PCA
  - t-SNE (t-distributed stochastic neighbor embedding)

- Detektim të anomalive apo defekteve
  - Apriori
  - Eclat (Equivalence Class Clustering and bottom-up Lattice Traversal)

3.2.3 Reinforcement learning

Teknika e reinforcement learning ka si bazë zgjidhjen e problemit të gjetjes së veprimeve të përshtatshme që duhen marrë në një situatë të caktuar në mënyrë që të rritet (maksimizohet) një shpërblim [10]. Në këtë kontekst modeli (learning system) merr formën e një agjenti sikurse tek Fig. 5 i cili vrojton ambientin dhe pastaj zgjedhë dhe kryen veprime të nevojshme në mënyrë që të shpërblehet [8]. Nëse veprimet që ndërmerr agjenti e largojnë atë nga qëllimi, atëhere ai merr dënim (penalty). Në këtë mënyrë, sistemi apo agjenti kalibrohet ashtu që në të ardhmen të dijë të marrë vendimin e duhur qe e afron atë tek qëllimi kryesorë. Në rastin e AlphaGo[4], qëllimi kryesorë është arritja e fitorës në lojën e famshme Go. AlphaGo fillimisht ka mësuar të luaj duke vëzhguar afro 30 milion lëvizje nga ekspertë të njohur të lojës [12], e pastaj ka mësuar duke luajtur kundër një instance tjetër të vetvetës.

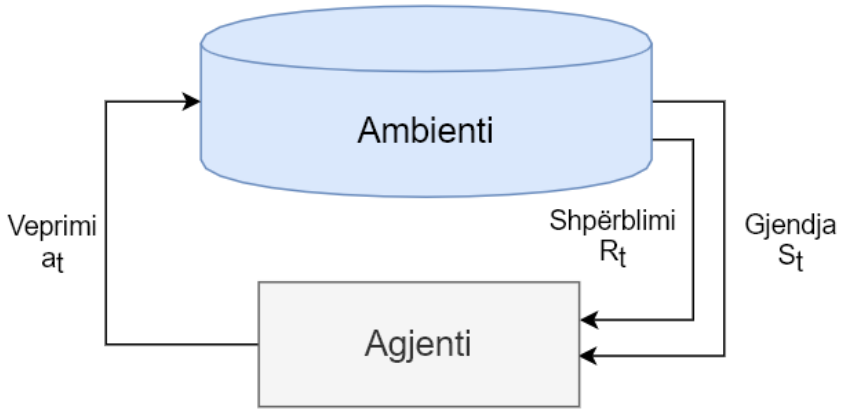

Fig. 5: Skema e funksionimit të një sistemi ML në principin e reinforcement learning [13]

[4] AlphaGo është një program kompjuterik që luan lojën Go.

3.3 Rrjetat artificiale neurale

Rrjetat artificiale neurale (Artificial Neural Networks – ANN) bëjnë pjesë në grupin e algoritmeve të Deep Learning[5]. Ideja për këto lloj struktura është huazuar nga vetë natyra, nga sistemi nervor biologjik i njeriut apo më saktësisht në njësinë themelore të sistemit nervor e cila është neuroni. Termi "rrjetë neurale" e ka origjinën nga përpjekjet për të gjetur paraqitje matematike të procesimit të informacionit në sistemet biologjike [14]. Rrjetat neurale artificiale sikurse në Fig. 6 ofrojnë zgjidhje për problemet më komplekse siç janë klasifikimi i miliarda fotografive (Google Images), njohja e zërit (Apple Siri) ose rekomandimi i videove për miliona shikues (Youtube) [8].

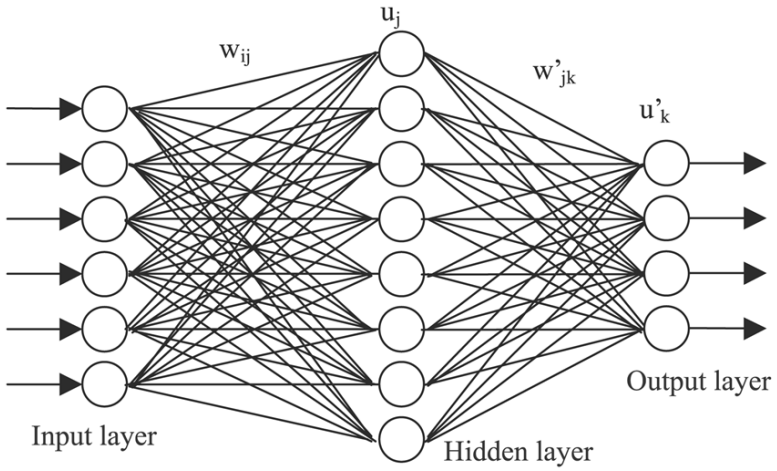

Fig. 6: Ilustrim i një strukture të një rrjete neurale artificiale [15]

Duke u bazuar në strukturën e neuronit biologjik, është derivuar struktura e *përceptronit q*ë paraqet arkitekturën më të thjeshtë të një rrjete artificiale neurale. Në vijim shpjegohet shkurtazi struktura dhe funksioni i përceptronit.

Për herë të parë u zbulua nga Frank Rosenblatt në vitin 1958 [16]. Struktura e përceptronit që është paraqitur tek Fig. 7 është e thjeshtë dhe në parim përbëhet nga katër pjesë kryesore:

- Vlerat hyrëse ose shtresa e hyrjeve (Input layer)
- Peshat ose lidhjet (Weights)
- Animi[6] (Bias)
- Funksioni i aktivizimit (Activation function)

Shpjegimi i përceptronit është me rëndësi kritike, pasi në të njejtën mënyrë dhe sigurisht me kompleksitet të shtuar do të funksionojnë edhe tipet tjera të rrjetave artificiale neurale. Funksionimi i përceptronit më së lehti sqarohet përmes një shembulli konkret.

[5] Deep Learning është nënfushë e Machine Learning që bazohet kryesisht në rrjetat neurale artificiale
[6] Termi *anim* do të përdoret në gjuhën angleze si *bias* në vijim gjatë ofrimit të sqarimeve

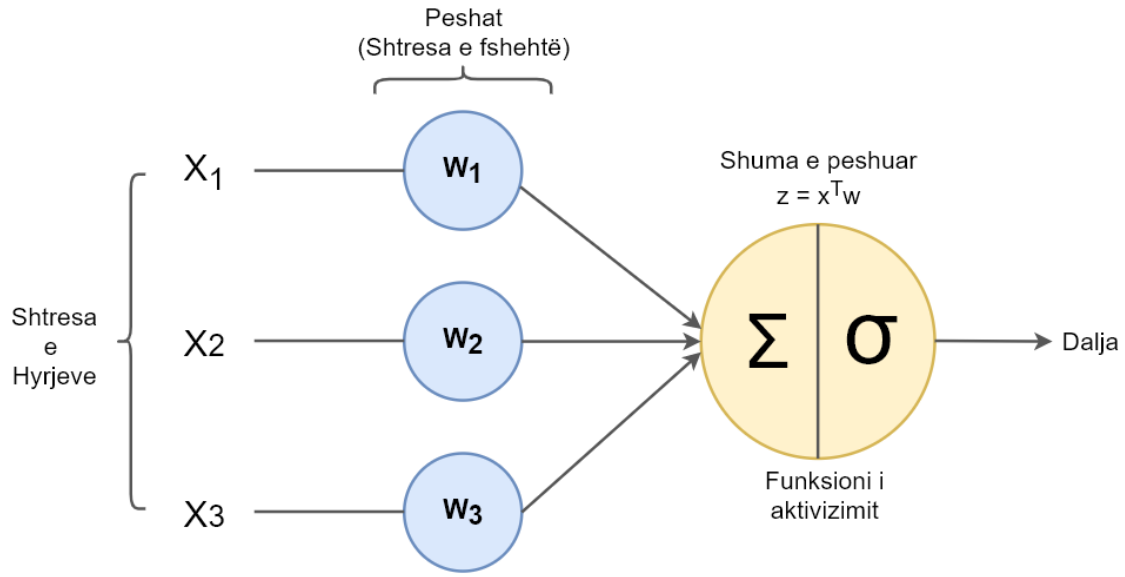

Fig. 7: Struktura themelore e përceptronit

Supozojmë se kemi vlerat hyrëse $\mathbf{X_1}$, $\mathbf{X_2}$ dhe $\mathbf{X_3}$ në shtresën hyrëse (input layer). Fillimisht të gjitha vlerat hyrëse do të shumëzohen me peshat (weights) përkatëse $\mathbf{W_1}$, $\mathbf{W_2}$ dhe $\mathbf{W_3}$, prej nga fitohet shuma e peshuar **z** (weighted sum):

$$\boldsymbol{z = \sum_{i=1}^{i=3} (X_i * W_i)}$$

Shuma e fituar i nënshtrohet një funksioni të aktivizimit që kryen detyrën e pasqyrimit të shumës në një vlerë brenda një intervali të caktuar, zakonisht (0,1) ose (-1, 1). Në praktikë shpesh përdoren funksioni shkallë, tanh, sigmoidi etj. siç është paraqitur tek Fig. 8. Pas këtij procesi, në dalje do të paraqitet një vlerë e kufizuar (nëse paraprakisht është aplikuar funksioni i aktivizimit). Në këtë gjendje siç u paraqit, përceptroni mund të konsiderohet si një klasifikues binar[7].

Peshat përcaktojnë fuqinë e secilës nyje, pra përcaktojnë se sa ndikim do të ketë në rezultatin dalës njëra prej hyrjeve. Vlerat në shtresën hyrëse në praktikë konsiderohen si një matricë dhe quhen *features* (sq. veçori). Procesi ku vlerat ushqehen në sistem dhe pas kryerjes së operacioneve të nevojshme rezultohet në një vlerë dalëse quhet *feed forwarding* dhe është koncept kyç në fushën e rrjetave neurale artificiale.

[7] Supozojmë vlerën dalëse të kufizuar në mes të 0 dhe 1, atëhere klasifikimi mund të bëhej në formë ashtu që vlera 1 të përcaktoj probabilitet maksimal se vlera hyrëse i takon kategorisë së parë ose të dytë (varësisht nga konfigurimi).

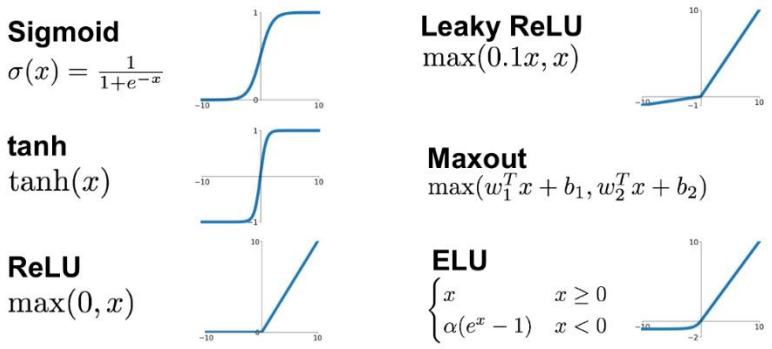

Fig. 8: Funksionet e aktivizimit të përdorura shpesh në praktikë [17]

Vetë rrjetat artificiale neurale të tilla quhen rrjeta *feed-forward* dhe veçohen nga karakteristikat si në vijim [18]:

- Përceptronët radhiten në formë të shtresave sikurse në Fig. 9 ku shtresa e parë i përmbanë hyrjet (inputet) kurse shtresa e fundit i përmbanë daljet (outputet). Shtresat në mes nuk kanë lidhje me pjesët e jashtme, dhe si të tilla quhen shtresa të fshehta (hidden layers)
- Secili përceptron në një shtresë lidhet me secilin përceptron të shtresës tjetër nga edhe ku është derivuar shprehja *feed-forward*
- Nuk ekzistojnë lidhje në mes të përceptronëve që i takojnë shtresës së njejtë

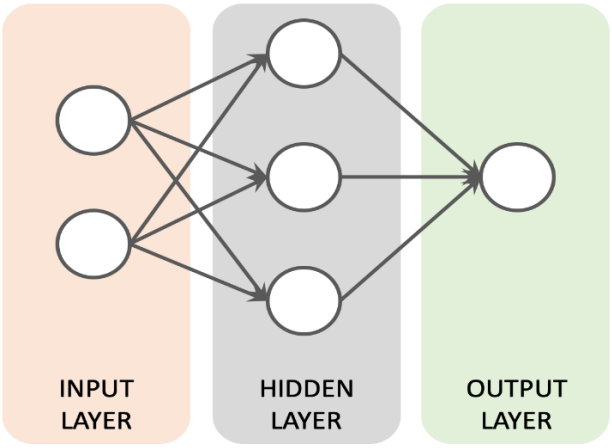

Fig. 9: Shtresat e një rrjete neurale feed-forward [19]

Funksionimi i rrjetave neurale artificiale mbështetet në principin se për çfarëdo hyrje (input) pritet që në dalje të paraqitet vlera e përafërt e saktë. Për temën që do të trajtohet në punim, pritet që rrjeta neurale artificiale për një email që ushqehet në hyrje, në dalje të nxjerrë një probabilitet se a është email në fjalë spam ose jo spam. Saktësia e shtresës në dalje varet plotësisht nga struktura e rrjetës neurale artificiale përfshirë këtu edhe vlerën e peshave (weights) dhe bias për secilën njësi të secilës shtresë. Procesi i mësimit apo i trajnimit për rrjetat neurale artificiale quhet *backpropagation* dhe

në shumicën e rasteve i takon kategorisë së mësimit të mbikqyrur (supervised learning) [18]. Trajnimi bëhet ashtu që modelit i paraqiten hyrje për të cilat dihet dalja e saktë dhe pas secilit iterim rregullohen vlerat e peshave dhe bias, ashtu që modeli të vendosë daljen e kërkuar. Pas procesit të trajnimit modeli duhet të jetë në gjendje që kur të përballet me hyrje të panjohura të mund të parashikoj daljen përkatëse. Në rastin e klasifikimit të spam email, modelit i prezentohen shumë çifte email dhe etiketë (spam ose jo spam), dhe pasi të ketë mësuar duhet të jetë në gjendje që për një email që nuk ka parë asnjëherë të përcaktoj klasifikimin përkatës. Pas çdo cikli të mësimit llogaritet gabimi i daljes e pastaj përdoret për ndryshimin e vlerave të peshave dhe bias varësisht nga nevoja (rriten ose zvogëlohen).

3.4 Hash algoritmet

Edhe pse hash algoritmet ose hash funksionet nuk lidhen në ndonjë formë direkt me fushën apo teorinë e Machine Learning, një trajtim i shkurt i tyre është i nevojshëm për shkak se mekanizmi për reduktimin e spam emailave i cili do të prezentohet më vonë gjatë punimit përdorë gjerësisht këto lloj të algoritmeve. Hash funksionet janë element kyç i fushës së kriptografisë. Hash funksionet pranojnë në hyrje një të dhënë dhe si rezultat në dalje vendosin një vlerë (bit-string) me gjatësi fikse sikurse është ilustruar tek Fig. 10. Kjo vlerë e llogaritur quhet ndryshe *digest* ose hash vlerë dhe mund të konsiderohet si *fingerprint* i një mesazhi, pra një reprezentim unik i mesazhit [20].

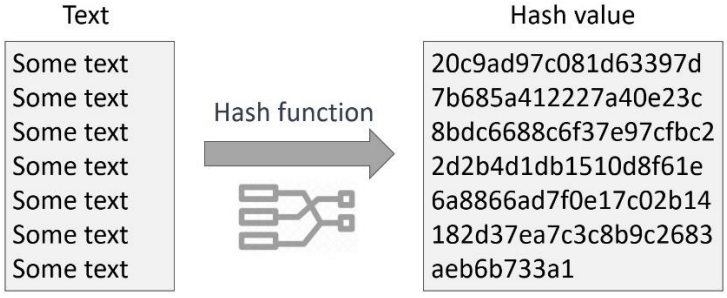

Fig. 10: Gjetja e hash vlerë së një të dhëne [21]

Hash funksionet gjejnë zbatim të madh në sigurinë kibernetike ku mundësojnë sesionet, ruajtjen e fjalëkalimeve, verifikimi e integritetit të të dhënave etj. Në sistemin që do të propozohet në pjesët tjera të tekstit, hash funksionet do të shfrytëzohen për aplikimin e konceptit të vërtetësisë së punës e njohur si *proof of work*. Një veti shumë e dëshirueshme e hash funksioneve është rezistenca ndaj dukurisë të quajtur *hash collision*. Për një hash funksion thuhet se vuan nga kjo dukuri nëse ekzistojnë të paktën dy hyrje (të dhëna) të ndryshme që gjenerojnë të njejtën hash vlerë në dalje. Në vijim përmenden disa nga hash funksionet më të njohura dhe përdorura:

- **SHA-1** (Secure Hash Algorithm 1) – Ka gjatësi të *digest* prej 160 bit, nuk konsiderohet më i sigurtë por përdoret ende
- **SHA-256** dhe **SHA-512** – Pjesë të familjes të SHA-2, të ngjashme me SHA-1. Kanë gjatësi të *digest* prej 256 dhe 512 bit respektivisht, sot përdoren gjerësisht
- **bcrypt** – Ka gjatësi të *digest* prej 184 bit, përdoret shumë për ruajtjen e fjalëkalimeve

4 Rasti i studimit – detektimi dhe reduktimi i spam emailave

4.1 Analiza e zgjidhjeve aktuale

Duke u bazuar në shpjegimet që janë dhënë përgjatë punimit, janë evidente ndikimi negativë dhe përmasat e dukurisë së dërgimit të spam email. Si dukuri daton shumë herët dhe si pasojë është investuar shumë kohë dhe resurse në gjetjen e mekanizmave të përshtatshëm për detektimin e spam emailave. Sistemet e para që u krijuan me qëllim të detektimit të spam emailave shfrytëzuan analizën statistikore të tekstit. Nëse marrim për bazë gjuhën angleze, duke nxjerrë histograme dhe vizualizime për fjalët e përdorura në një spam email, është zbuluar se shumica e spam emailave kanë të përbashkët një grumbull fjalësh. Një vizualizim i tillë është paraqitur tek Fig. 11:

Fig. 11: Fjalë të shpeshta të përdorura në spam email në gjuhën angleze [22]

Duke përdorur këtë informacion u krijuan mekanizma që analizonin tekstualisht trupin e një emaile për detektimin e këtyre fjalëve. Nëse përmbajtja tekstuale e email posedon një tepricë të njërës apo më shumë nga fjalët kyçe (keywords), atëhere email në fjalë do të kategorizohej si spam dhe tek marrësi do të paraqitej në shportë (junk).

Edhe pse kjo metodë u konsiderua si efikase, shumë lehtë dhe shpejtë u gjetën metoda për të anashkaluar dhe mashtruar këtë lloj të detektimit. Njëra prej formave ishte përdorimi i teknikave të obfuskimit[8] të fjalëve kyçe indikatorë të spam [23]. Si shembull, fjala *"falas"* mund të ndahet në copëza me vija ndarëse dhe të shndërrohet në "*-f-a-l-a-s"* dhe si e tillë do të anashkalohej nga shumica e mekanizmave detektues [22]. Në mënyrë që të luftohej dukuria e obfuskimit, mekanizmat detektues evoluan dhe shfrytëzuan teknika të de-obfuskimit. Krijimi i metodave për detektim mori një kahje tjetër kur filloi zbatimi i teknikave dhe algoritmeve të Machine Learning. Njëra prej metodave kryesore ishte teorema e Bayesit e probabilitetit kondicional. Për herë të parë kjo metodë u prezentua nga kompania Microsoft në vitin 1998 me një sistem të quajtur filtri Bayesian (Bayesian filter) [24]. Këto sisteme shfrytëzonin analizën e një numri të madh të emailave të kategorizuara si emaila të besuara ose spam. Sistemi fillimisht konfigurohej me disa fjalë kyçe indikatorë të spam e pastaj më vonë kjo listë rritej edhe më shumë duke pranuar raportime nga përdorues të email. Logjika e këtyre sistemeve bazohet në përdorimin e formulës gjenerale të Bayes-it për probabilitet. Duke ditur probabilitetin e paraqitjes në email të një fjale kyçe që është indikatorë i spam, probabilitetin që një email të jetë spam dhe probabilitetin që një spam email të

[8] Në kontekstin e sigurisë në Internet *obfuskimi* nënkupton praktikën e të vështirësuarit të kuptimit të një të dhëne

përmbajë këtë fjalë kyçe, mund të llogaritet probabiliteti që emaila në fjalë është spam (junk). Një sistem kompjuterik shumë i njohur për filtrimin e spam emailave është *Apache SpamAssassin* që për herë të parë u paraqit në prill të vitit 2001 dhe sot (në kohën kur është shkruar ky punim) mbanë titullin si programi numër një me kod burimor të hapur (open source) për filtrim të spam emailave në ambientet ndërmarrëse (enterprise) [25]. Njëra prej metodave të shumta për detektim që përdoret nga SpamAssassin është edhe vetë filtrimi Bayesian. Sistemet e filtrimit që bazohen në këtë koncept kanë dobësi sulmin e quajtur helmimi Bayesian (Bayesian poisoning). Sulmi funksionon ashtu që para se të dërgohet spam emaila, ajo injektohet me fjalë të ndryshme të zakonshme me shpresë se analiza statistikore dhe probabilistike do të dështojë. Në konferencën CEAS të vitit 2004 u prezentua pikërisht ky lloj i sulmi i injektimit të fjalëve të zakonshme në një spam email [26]. Shpesh ky sulm u quajt joefektiv sepse për të mundur peshën statistikore që shkakton qoftë edhe vetëm një fjalë kyçe (indikatorë i spam), duhet të shtohen dhjetra fjalë tjera të rëndomta dhe të zakonshme duke përfshirë emra të personave apo termeve tjera [22]. Sot, në qarkullim ekzistojnë mekanizma për detektim dhe filtrim të spam email të bazuar në Machine Learning dhe Deep Learning si nënfushë të saj. Kompania Google në vitin 2015 deklaron se mekanizmat që përdorë për filtrimin e spam emailave në shërbimin Gmail kanë saktësi të filtrimit deri në 99.9% me shkallë të gabimit[9] vetëm 0.05% [27]. Për arritjen e kësaj saktësie Google përdorë rrjetat neurale artificiale. Duke marrë parasysh se në ditët e sotme përmbajtja e emailave nuk është më thjeshtë tekstuale por përmbanë edhe imazhe dhe shtesa tjera si dokumentet (attachments), detektimi i spam emailave nuk duhet të bazohet vetëm në analizën e përmbajtjes tekstuale. Prandaj, një mekanizëm ideal për detektim të spam emailave do të bënte analizën e të gjitha këtyre komponenteve për nxjerrjen e një parashikimi se cilës kategori i takon emaila, spam ose jo spam.

4.2 Metodologjia dhe veglat e përdorura

Si për çdo problem tjetër, forma apo mënyra se si i qasemi problemit është me rëndësi kritike pasi që drejtpërdrejtë ndikon në rezultatin dhe kualitetin e zgjidhjes. Pas përcaktimi të problemit menjëherë janë vendosur hapat që duhen përcjellur deri në arritjen tek zgjidhja. Duke pasur parasysh që për problemin e detektimit apo klasifikimit të spam emailave është vendosur që të shfrytëzohen algoritme të fushës së Machine Learning respektivisht rrjetat neurale artificiale, atëherë deri në arritjen e zgjidhjes duhen konsideruar (ndjekur) patjetër hapat në vijim:

1. Grumbullimi i të dhënave
2. Përpunimi i të dhënave
3. Përcaktimi i arkitekturës së modelit (numri dhe lloji i shtresave)
4. Trajnimi i modelit
5. Vlerësimi i performancës së modelit

Për fazën e implementimit është përcaktuar që programi kompjuterik të shkruhet në gjuhën programuese Python 3. Arsyeja më e fortë që dërgon tek zgjedhja e Python si vegël apo gjuhë programuese për konstruktimin e programeve që kanë të bëjnë me Machine Learning apo inteligjencës artificiale në përgjithësi është numri jashtëzakonisht i madh i librarive që janë në

[9] Në këtë kontekst, gabim konsiderohet klasifikimi i emailave të nevojshme (të pastra) në spam edhe pse nuk i takojnë asaj kategorie.

dispozicion sikurse TensorFlow, Scikit-learn, Keras etj. Këto janë librari ose korniza (frameworks) të gatshme për implementimin e algoritmeve të Machine Learning. Vlenë të përmendet edhe libraria e njohur NumPy që shërben për llogaritje shkencore të shpejta dhe analizim të të dhënave. Implementimi i Python 3 nuk shquhet për performancë të lartë, sidomos në krahasim me disa gjuhë tjera programuese [28]. Ky fakt nuk paraqet aspak problem në implementimin e zgjidhjes tonë që përfshinë rrjetat neurale artificiale dhe që për funksionim kërkojnë miliona operacione matematikore me numra me presje dhjetore (floating point numbers) dhe matrica. Arsye për këtë është se shumica e këtyre veprimeve që kryhen përmes librarive sikurse NumPy janë të implementuara në gjuhë programuese të ulëta (sikurse C, C++ ose Fortran), dhe shpesh implementohen direkt në nivel të makinës (assembly). Ndërkohë, për Python shkruhen vetëm librari mbështjellëse (wrappers) që bëjnë thirrjen e funksionalitetit të implementuar një nivel më poshtë. Për arritjen deri tek zgjidhja e problemit që shtron punimi, është përdorur gjerësisht libraria Keras[10] (versioni 2.3.1) e cila është librari me kod burimor të hapur (open source) e shkruar me qëllim kryesorë për implementimin e rrjetave neurale artificiale. Zgjidhja është zhvilluar dhe testuar në versionin 3.7.5 të Python 3, në një makinë virtuale që punon me sistemin operativ Ubuntu (versioni 19.10) të familjes Linux. Për trajnim, makina virtuale ka pasur në dispozicion tetë gigabajt memorie (RAM), procesor i7 të gjeneratës së katërt, ndërsa nuk ka qenë në dispozicion një kartelë grafike, pavarsisht pse ky resurs do të shpejtonte procesin në tërësi.

4.3 Detektimi i spam emailave përmes rrjetave neurale LSTM

4.3.1 Grumbullimi i të dhënave

Është përmendur shpesh përgjatë punimit se për trajnimin e rrjetës neural artificiale (në vazhdim referohet si model) nevojitet një grumbull shumë i madh i të dhënave. Sigurisht në kontekstin e temës që po trajtohet, grumbulli i të dhënave ose dataseti do të jetë një grumbulli i emailave reale të dërguara dhe pranuara. Në këtë grumbull duhet që të kemi një përzierje të emailave të kategorisë spam dhe jo spam. Të shohim në vijim disa fusha të shkëputura nga struktura e header-it [22] të një email të dërguar përmes SMTP[11] sikurse tek Fig. 12.

- **Received –** informata për rrugën që ka kaluar email, serverët, datat, IP adresat etj.
- **From –** email adresa e dërguesit
- **To –** email adresa e marrësit
- **Return Path –** një adresë opsionale në rast se email dështon të mbërrijë në destinacion
- **Message ID -** një vlerë identifikuese unike e përcaktuar nga sistemi elektronik postar
- **X-mailer -** programi që është përdorur për të krijuar dhe dërguar emailën
- **Subject –** një mesazh i shkurtër që ka për qëllim përmbledhjen e thelbit të emailës

[10] Për Keras mund të mësoni më shumë në lidhjen keras.io
[11] SMTP – Simple Mail Transfer Protocol është protokoll komunikimi për shkëmbimin e emailave

```
Received: from mailer.abc.uk ([XXX.XXX.XXX.XX2]) by
  mailtr.an.organisati.on with esmtp (Exim 4.80) id 1Y25L6-0002zV-oA
  for joe.monday@an.organisati.on; Sun, 14 Dec 2014 18:30:01 +0000
Received: from [XXX.XXX.XXX.XX1] (helo=[192.168.1.8]) by mailer.abc.uk
  (Postfix) with ESMTPSA (envelope-from <john.smith@abc.uk>) id
  2312312132; Sun, 14 Dec 2014 18:30:00 +0000
Content-Type: multipart/alternative;
  boundary="Apple-Mail-QWNHJDNF-KJDD"
MIME-Version: 1.0 (1.0)
Subject: Information on the purchase order
From: John Smith <john.smith@abc.uk>
X-Mailer: iPad Mail (12B440)
Date: Sun, 14 Dec 2014 18:30:05 +0000
CC: Pat Tuesday <pat.tuesday@an.organisati.on>
Content-Transfer-Encoding: 7bit
Message-ID: <KJDKUJSK-EOI-K2LS-DJFD-KJSD49SEI@abc.uk>
To: Joe Monday <joe.monday@an.organisati.on>
X-Username: jksmith3
Return-Path: john.smith@abc.uk
```

Fig. 12: Shembull i strukturës së header-it të një emaile të derguar [29]

Sigurisht se pas header-it të emailës do të vijojë dikur edhe përmbajtja tekstuale e emailës e cila ndryshe quhet trupi i emailës (email body). Është pikërisht kjo pjesë e emailës në të cilën do të fokusohet problemi i zgjedhur në këtë punim. Sigurisht se një program i sofistikuar për detektim dhe filtrim të spam emailave do të merrte në konsiderim çdo copëz të emailës. Një program i tillë do të analizonte email adresën e dërguesit, do të konfirmonte se mos ndoshta kjo email adresë i takon një liste të zezë me email adresa të dyshimta. Pastaj mund të analizohen edhe copëza tjera sikurse tipi i të dhënave (Content-Type) dhe sigurisht edhe subjekti i emailës së dërguar. Por në këtë punim vëmendja është përqëndruar plotësisht në analizën e trupit të emailës përmes rrjetave neurale LSTM. Si rrjedhojë, në datasetin që do ta përdorim për trajnimin e modelit do të na mjaftojë të kemi çifte të përmbajtjes tekstuale të emailës (email body) dhe etiketimit përkatës sikurse në Fig. 13.

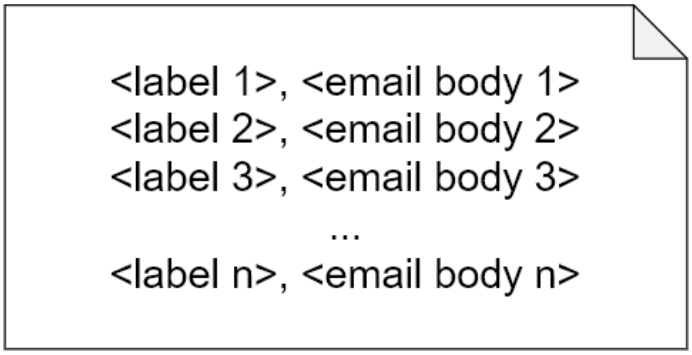

Fig. 13: Struktura e dëshiruar e dataset-it për trajnim të modelit

Idealisht, ky dataset do të përfshinte çiftet e ndara me presje në formatin CSV. Për shkak se nuk ekziston një dataset definitiv me email të dërguara në gjuhën shqipe, është marrë vendimi që në këtë punim programi final që detekton spam emailat do të funksionoj vetëm me emaila teksti i të cilave është shkruar në gjuhën angleze. Për shkak se në front të qëllimit të punimit është demonstrimi dhe shpjegimi i mekanizmave detektues të spam emailave dhe jo krijimi i një produkti final për përdorim, kufizimi vetëm në gjuhën angleze e jo atë shqipe nuk përbën ndonjë pengesë.

Si përfundim, një dataset publik i cili përmbushë të gjitha kushtet e parashtruara është publikuar në vitin 2011 [30] me titullin *"SMS Spam Collection Data Set"* dhe përmbanë rreth 5,500 përmbajtje tekstuale me gjatësi të ndryshme të etiketuara si spam ose jo spam (ham). Ky është një dataset që mund të përdoret për trajnim dhe testim të zgjidhjes së ofruar nga punim. Por, programi final i zhvilluar ofron fleksibilitet që të trajnohet me çfarëdo dataseti për sa kohë është në përputhje me formatin e përcaktuar.

4.3.2 Përpunimi i të dhënave

Dateseti që u përcaktuar për shfrytëzim edhe pse përmbanë formatim të thjeshtë, nuk është i gatshëm që menjëherë të ushqehet në model për trajnim pasi ai duhet përpunuar paraprakisht. Fillimisht, struktura e të dhënave në dataset ka formën ashtu që etiketa është e ndarë nga teksti me një hapsirë. Kjo përbën pengesë minimale, pasi lehtësisht mund të strukturohet në program në dy lista të ndara pasi që të jetë lexuar dataseti nga fajlli sikurse tek Kodi 1.

```
def load_data_from_file(path: str) -> tuple:
    with open(path, 'r') as spam_collection_file:
        labels = []
        emails = []
        for line in spam_collection_file:
            label, email = line.strip().split(maxsplit=1)
            labels.append(label.strip())
            emails.append(email.strip())

    return labels, emails
```

Kodi 1: Shembull i një copëze të kodit për përpunimin e datasetit në fjalë

Përpunimi i të dhënave nuk përfundon në këtë pikë. Nëse vështrojmë etiketat (labels) tek dataseti, vërejmë se kategoritë janë të shënuara me fjalën *"spam"* kur teksti është spam dhe me *"ham"* kur teksti nuk cilësuar si spam. Kjo formë e reprezentimit nuk është e përshtatshme pasi për ta mësuar rrjetën neurale me njësi LSTM na duhen vlera numerike. Një formë e thjeshtë por që gjenë zbatim shumë në praktikë është teknika one-hot encoding që është paraqitur në vijim tek Fig. 14.

$$\text{etiketa}[n] = \begin{cases} [1, 0] \text{ nëse "ham"} \\ [0, 1] \text{ nëse "spam"} \end{cases}$$

Fig. 14: Konvertimi i etiketave përmes one-hot encoding

Për rastin e klasifikimit binar, pra në dy kategori, secila etiketë shndërrohet në një vektorë me madhësi dy. Vendoset një konventë ose rregull arbitrare e cila mund të përcaktoi se etiketa "*ham*" (jo spam) është e barabartë me vektorin [1, 0] ndërsa etiketa "*spam*" është e barabartë me vektorin [0, 1]. Pra pozita e parë në vektorë shërben si indikatorë se teksti është jo spam nëse vlera në atë pozitë është 1, e njejta vlenë edhe për pozitën dy. Konventa se cila pozitë shënon cilën etiketë është arbitrare, por me rëndësi të madhe është që pasi të caktohet një konventë, ajo të përcillet deri në fund. Me rëndësi është që për një vektorë me gjatësi **n**, vetëm njëri nga anëtarët e tij të ketë vlerën 1, nga edhe është derivuar emri i metodës one-hot encoding.

Hapi i radhës është përpunimi i pjesës tekstuale të datasetit. Është evidente se rrjeta neurale me njësi LSTM si hyrje në shtresën e hyrjes do të pranojë vlera numerike, gjegjësisht vektorë me numra që më herët i kemi quajtur si *features*. Prandaj, pjesët tekstuale duhen shndërruar në vlera numerike. Së pari do të bëhet pastrimi i fjalëve nga fragmentet e ndryshme (karakteret) të padëshiruara siç janë pikësimi dhe shenjëzimet e ndryshme. Funksionaliteti i kërkuar është i implementuar në klasat ndihmëse për procesim të tekstit në librarinë Keras. Klasa që do të shfrytëzohet është klasa Tokenizer [31]. Tokenizer do të kryej pastrimin e secilës fjalë dhe do të gjeneroj një fjalor të fjalëve me indeksat përkatës. Indeksat përcaktohen në bazë të shpeshtësisë së fjalës në fjali ashtu që fjalët më të shpeshta marrin indeksat më të vegjël. Nëse marrim një listë të shkurtë me përmbajtje tekstuale të çfarëdoshme dhe aplikojmë funksionin "*fit_on_texts*" të Tokenizer, do të gjenerohet fjalori me indeksat (word index) si tek Kodi 2.

```
test_emails = [
    "This is a sample email!",
    "This is another sample email!"]

tokenizer = Tokenizer()
tokenizer.fit_on_texts(test_emails).word_index
```

```
    {'this': 1, 'is': 2, 'sample': 3, 'email': 4, 'a': 5, 'another': 6}
```

Kodi 2: Coptimi, filtrimi dhe indeksimi i fjalëve përmes Tokenizer

Pasi të gjenerohet fjalori me indeksa (word index) mund të aplikohet funksioni "*text_to_sequences*" i klasës Tokenizer që do të transformoj fjalët e secilit tekst në indeksa përkatës sikurse tek Kodi 3.

```
[[1, 2, 5, 3, 4], [1, 2, 6, 3, 4]]
```

Kodi 3: Transformimi i fjalëve në indeksa përmes Tokenizer

Menjëherë shtrohet pyetja se çfarë gjatësie do të jenë vektorët që do të krijohen. Gjatësia e këtyre vektorëve do të jetë fikse dhe ajo mund të përcaktohet nga shfrytëzuesi. Në rastet kur teksti është më i shkurtër se gjatësia fikse e vektorit, pozitat boshe të vektorit do të mbushen me vlerën zero sikurse tek Kodi 4, kjo sepse ato duhen definuar patjetër para se të përdoren për trajnimin e modelit.

```
[[   0    0    0 ...   58 4436  144]
 [   0    0    0 ...  472    6 1940]
 [   0    0    0 ...  660  392 2997]
                ...
 [   0    0    0 ...  107  251 9008]
 [   0    0    0 ...  200   12   47]
 [   0    0    0 ...    2   61  268]]
```

Kodi 4: Struktura e të dhënave pas procesimit

Dataseti është pothuajse i gatshëm për tu ushqyer tek modeli për trajnim, por paraprakisht duhen marrë edhe disa hapa tjerë. Njëri prej hapave që duhet ndërmarrë është edhe ndarja apo fragmentimi i të dhënave të përpunuara sikurse në Fig. 15. Ndarja bëhet në dy grupe të pabarabarta:

- Të dhënat për trajnim
- Të dhënat për testim

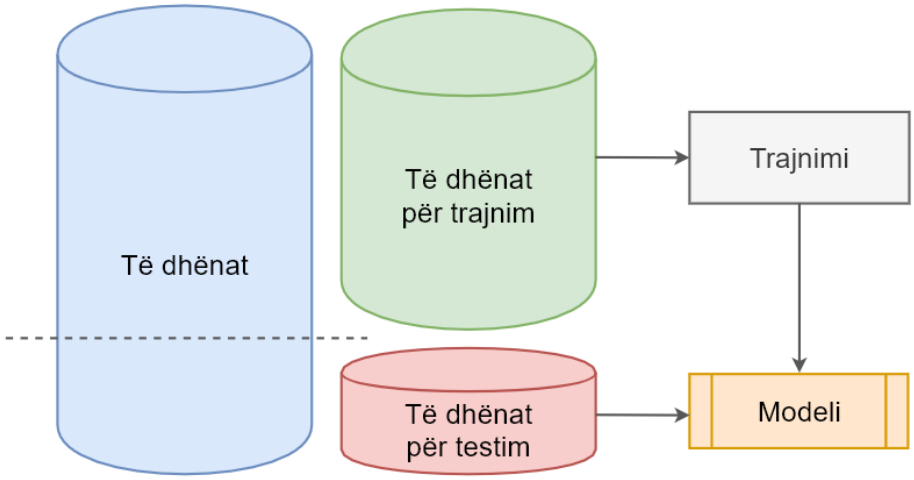

Fig. 15: Skema e ndarjes së të dhënave në grupin për trajnim dhe testim [32]

Ndarja vlenë si për etiketat (labels) ashtu edhe për tekstet. Si rrjedhojë, nga dy grupe që përmbanin etiketat dhe tekstet, tash do të krijohen katër grupe:

- Grupi i etiketave (labels) për trajnim
- Grupi i teksteve për trajnim
- Grupi i etiketave (labels) për testim
- Grupi i teksteve për testim

Kuptohet nga vetë emërimi i grupeve se njëra pjesë do të përdoret për ushqimin e modelit gjatë fazës së trajnimit e një pjesë tjetër do të lihet anash për fazën e testimit nga e cila do të bëhet matja e performancës së modelit. Edhe pse ndarjen e të dhënave mund ta programojmë vetë, do të shfrytëzojmë funksionalitetin që na ofron libraria scikit-learn.

Kjo librari përmbanë funksionin "*train_test_split*" i cili na mundëson ndarjen e të dhënave në raport sipas dëshirës siç demonstrohet tek Kodi 5.

```
texts_train, texts_test, labels_train, labels_test =
    train_test_split(texts, labels, random_state=10, test_size=0.25)
```

Kodi 5: Ndarja e të dhënave të përpunuara në grupin për trajnim dhe testim

Të dhënat do të ndahen saktësisht sipas raportit të përcaktuar në parametrin "*test_size*" dhe në mënyrë të rëndomtë. Parametri "*random_state*" është një vlerë numerike e cila shërben si farë (s*eed*) që përdoret nga gjeneratori i numrave të rëndomtë gjatë seleksionimit të rëndomtë të të dhënave për ndarje. Është me rëndësi që kjo vlerë të ruhet nëse në të ardhmen nevojitet të krijohet ndarja saktësisht e njejtë.

Si përfundim, mund të përmblidhen hapat që u morën për përgatitjen e të dhënave për trajnim dhe testim të modelit:

1. Leximi dhe parsimi i datasetit në dy lista, një që përmbanë etiketat (labels) dhe një tjetër që përmbanë pjesën teksuale për secilën email
2. Enkodimi i etiketave përmes metodës one-hot encoding
3. Krijimi i fjalorit të fjalëve me indeksat përkatës (word index) me ndihmën e Tokenizer
4. Konvertimi i të gjitha fjalëve në indeksa përkatës
5. Ndarja e të dhënave në grupin për trajnim dhe testim

4.3.3 Word embeddings – reprezentimi i shpërndarë i fjalëve

Për shkak të strukturës apo arkitekturës së rrjetës neurale artificiale që do të përcaktohet në vijim, trajnimi do të bëhet ashtu që modelit do ti prezentohen fjalët e tekstit një nga një. Si rrjedhojë kërkohet reprezentimi i fjalëve në formë të një vektori. Një metodë që mund ta shfrytëzojmë është përsëri metoda e one-hot encoding. Përdorimi i kësaj metode në këtë rast prodhon disa efekte anësore të papëlqyeshme. E para është se vektorët që krijohen për shenjëzimin e fjalëve do të kenë dimensione shumë të mëdha (varësisht nga numri total i fjalëve të veçanta që ekzistojnë në dataset). Së dyti, problem tjetër është se nga dy vektorë të ndryshëm që paraqesin dy fjalë të veçanta është vështirë të gjendet ndonjë operacion matematikor me vektorë ashtu që të mund të llogaritet ngjashmëria e dy fjalëve. Kjo më vonë do të ndikonte drejtpërdrejtë në rezultatet e modelit. Do të ishte ideale që të përdorej një formë e reprezentimit të fjalëve ashtu që secila fjalë të shprehej në formë të një vektori me një gjatësi të vogël e të kufizuar, dhe gjithashtu të ekzistonte një operacion matematikor i cili për dy vektorë të ndryshëm do të llogariste ngjashmërinë në mes atyre dy vektorëve, gjegjësisht fjalëve. Zgjidhjen për këtë dilemë e gjejmë tek *word embeddings*. *Word embeddings* janë një klasë e teknikave ku fjalët e veçanta paraqiten ose reprezentohen si vektorë me numra real në një hapsirë vektoriale të paradefinuar [33] sikurse në Fig. 16. Secilës fjalë i përgjigjet një vektorë përkatës me dimension të caktuar. Sigurisht, pasi vektorët janë shumë dimensional, për paraqitje grafike duhet aplikuar paraprakisht reduktimi i dimensionalitetit. Për

temën në trajtim e sipër paraqitja grafike e këtyre vektorëve nuk është e dobishme prandaj mjafton që të kemi reprezentimin e kërkuar para se të ushqejmë fjalët tek modeli që do të konstruktohet.

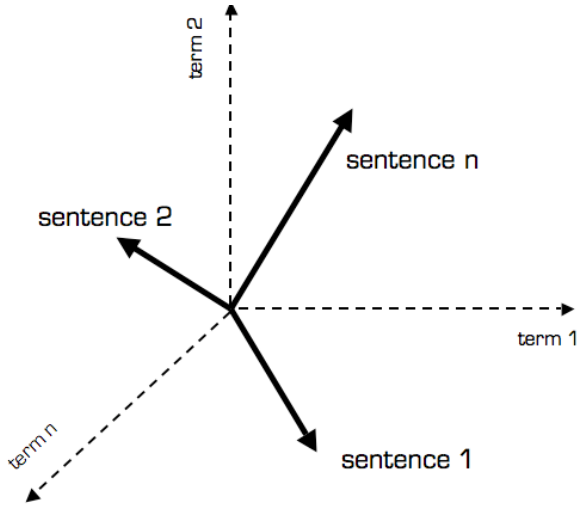

Fig. 16: Hapësira vektoriale dhe reprezentimi vektorial i fjalëve [34]

Për një gjuhë të caktuar sikurse është gjuha angleze që do të përdoret në këtë punim, ekzistojnë algoritme të gatshme dhe lista të gatshme për shumicën e fjalëve me reprezentimet vektoriale përkatëse. Disa prej tyre janë "Word2Vec" dhe "GloVe" [35]. Krahasimi i veçorive të njërës dhe tjetrës i takon më shumë fushës së procesimit të gjuhës natyrale (NLP) dhe është jashtë fushëveprimit të këtij punimi. Për ndërtimin e modelit në këtë punim do të përdoret "GloVe" për shkak të thjeshtësisë dhe fleksibilitetit që ofron në gjatësinë e vektorëve.

Sa për ilustrim, për një fjalë të caktuar si "university", reprezentimi vektorial nga "GloVe" me gjatësi 50 të vektorit do të ishte sikurse tek Kodi 6.

```
university = [-1.1082, 1.2916, -0.78751, -0.45955, ...,0.035944]
```

Kodi 6: Reprezentimi i shpërndarë i fjalës "university" përmes GloVe me madhësi 50

Në arkitekturën e rrjetës me njësi LSTM që do të jepet në vijim, *word embeddings* do të përdoren si parashtresë nëpër të cilën do të kalojnë dhe transformohen të gjitha fjalët para se të ushqehen tek pjesa tjetër e modelit. Pasi "GloVe" përmbanë me mijëra fjalë dhe padyshim shumë më shumë sesa ekzistojnë fjalë të veçanta në pjesën tekstuale të emailave të përpunuara të datasetit, nuk është e nevojshme që të ruhen në memorie reprezentimin e të gjitha fjalëve. Mjafton të ruhen vetëm ato reprezentime që paraqiten në dataset. Gjithashtu, pas filtrimit të listës, fjalët do të zëvendësohen me indeksat e tyre përkatës që u krijuan me ndihmën e Tokenizer, dhe si rrjedhojë do të kemi një fjalor me çifte (indeks, reprezentim vektorial) të gatshme për t'iu bashkangjitur modelit si shtresë në arkitekturën e tij. Në shtresën e *word embeddings* të arkitekturës së modelit do të përdoren vektorët e "GloVe" me gjatësi apo dimension 100. Kjo, njëherazi përcakton edhe numrin e hyrjeve në shtresën e hyrjeve të modelit.

4.3.4 Përcaktimi i arkitekturës së modelit

Siç u përmend shumë herë përgjatë punimit, për implementimin e modelit i cili do të bëjë klasifikimin e emailave do të shfrytëzohet libraria Keras. Kjo librari gjithashtu përmbanë implementime për rrjeta neurale artificiale sikurse që janë ato të tipit LSTM(Long Short Term Memory) që janë bërthama e zgjidhjes së problemit të shtruar. Në seksionin paraprak u cek se të gjitha fjalët së pari duhet të kalojnë nëpër shtresën e *word embeddings* (Fig. 17) për tu shndërruar në reprezentimet përkatëse vektoriale. Madhësia apo dimensioni i hyrjeve në shtresën Embedding do të jetë 100, aq sa është përcaktuar nga përzgjedhja e "GloVe". Po i njejti dimension do të jetë edhe në dalje të kësaj shtrese.

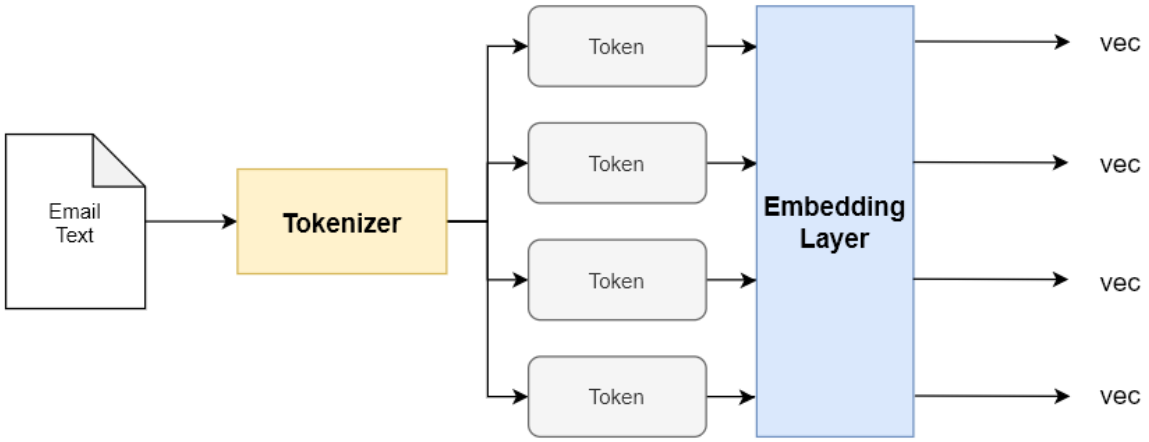

Fig. 17: Transformimi i tekstit në token nga Tokenizer e pastaj në vektorë

Hapi i radhës është që reprezentimet vektoriale të fjalëve të sapodala nga shtresa e *word embeddings* (Embedding Layer) të hyjnë në shtresën me rrjetën LSTM ku edhe kryhet puna kryesore për klasifikimin e tekstit të emailave. Rrjetat LSTM janë përzgjedhur duke u bazuar në faktin se janë shumë të përshtatshme edhe efektive në procesimin e të dhënave sekuenciale sikurse janë video regjistrimet, zëri apo në këtë rast përmbajtet tekstuale. Në shtresën e rrjetës neurale do të përdoren 128 njësi LSTM. Gjatë testimit me numër të ndryshëm të njësive, ky numër ka dhënë rezultatet më të kënaqshme. Arkitektura e modelit në përgjithësi do të dalloj gjithmonë varësisht nga dataseti, mënyra e përpunimit dhe qëllimi final që ka sistemi. Pas kapërcimit të shtresës me njësi LSTM do të kalohet në shtresën e rradhës të quajtur Dropout Layer. Kur të krijohet dhe përdoret, mund t'i përcaktohet një vlerë e quajtur *dropout rate* e cila specifikon probabilitetin që një hyrje në shtresë të barazohet me zero [36]. Shtrohet pyetja se pse pas gjithë kësaj pune të eliminohen disa vlera duke i kthyer ato në zero. Arsyeja është që të luftohet ose të paktën të zvoglohet dukuria e quajtur *overfitting*. Për një model të trajnuar thuhet se është nën ndikimin e *overfitting* kur modeli në fjalë performon shumë mirë në të dhënat që janë ndarë për trajnim por nuk performon aq mirë në të dhënat për vlersim ose testim [37]. Kjo ndodhë për arsye se modeli është akomoduar dhe ka memorizuar shumë mirë rastet nga të dhënat prej të cilave ka mësuar, dhe tash nuk është më në gjendje të gjeneralizojë kur i paraqiten raste (të dhëna) të reja. Shtimi i shtresës Dropout ka për qëllim regullimin e kësaj të mete dhe është njëra prej fomave më të thjeshta dhe të lira [36]. Në shtresën Dropout dimensioni i hyrjeve do të jetë 128 dhe po aq do të jetë dimensioni i daljeve. Në fund pasi që kemi të bëjmë me një klasifikues binar, pra klasifikim në dy kategori spam ose jo

spam, nënkuptohet se daljet e deritanishme duhet të komprimohen nga 128 në 2. Ky funksionalitet do të realizohet me shtimin e një shtrese tjetër dhe përfundimtare. Kjo shtresë quhet Dense Layer. Paraprakisht, daljeve do të i'u aplikohet një funksion i aktivizimi në mënyrë që vlerat dalëse të pasqyrohen në interval (0, 1) që paraqet edhe probabilitet. Funksioni i aktivizimit i përzgjedhur është *softmax*. Softmax përdoret në rastet kur një hyrje me **n** vlerave kërkohet të normalizohet në shpërndarje të probabilitet me **n** klasa të probabilitetit.

Si përfundim, arkitektura e plotë mund të paraqitet përmes skemës në Fig. 18:

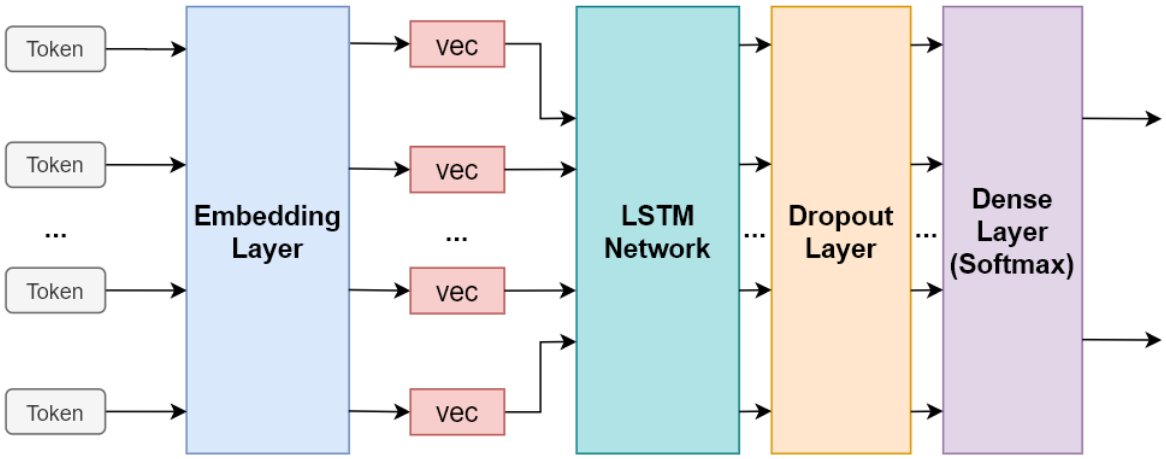

Fig. 18: Skema e arkitekturës së modelit të ndërtuar

Pasi tash arkitektura është e gatshme ajo shumë lehtë mund të implementohet në program përmes Keras. Fillohet me krijimin e shtresës Embedding sikurse tek Kodi 7.

```
embedding_layer = Embedding(len(embeddings_matrix),
                            100,
                            trainable=False,
                            100,
                            weights=[embeddings_matrix])
```

Kodi 7: Krijimi i Embedding Layer me klasën Embedding të Keras

Duke e përcjellur skemën e paraqitur më lartë vazhdohet me krijimin e shtresës që përmbanë rrjetën me njësi LSTM sikurse tek Kodi 8:

```
lstm_layer = LSTM(units=128, recurrent_dropout=0.25)
```

Kodi 8: Krijimi i shtresës apo rrjetës me LSTM njësi

Numri 128 tregon numrin e njësive LSTM që do të përdoren në këtë shtresë. Kjo shtresë mund të konsiderohet si shtresa më komplekse dhe kritike e të gjithë modelit. Në vazhdim, vendoset shtresa Dropout për të bërë rregullimin e dukurisë së Overfitting siç paraqitet tek Kodi 9:

```
dropout_layer = Dropout(rate=0.3)
```

Kodi 9: Krijimi i shtresës Dropout

Për fund, vendoset shtresa e quajtur Dense Layer që gjithashtu bën aplikimin e funksionit të aktivizimit softmax sikurse tek Kodi 10:

```
dense_layer = Dense(units=2, activation="softmax")
```

Kodi 10: Krijimi i shtresës Dense me funksionin e aktivizimit softmax

Gjatë sqarimit dhe nga skemat vërehet radhitja dhe natyra sekuenciale e modelit. Secila shtresë pranon hyrje dhe pas përpunimit bën përcjelljen në shtresën e radhës. Për këtë arsye edhe modeli i përzgjedhur quhet model sekuencial (Sequential). Tek Kodi 11 bëhet krijimi i modelit sekuencial në Keras dhe shtohen në mënyrë sekuenciale shtresat e krijuara më herët.

```
model = Sequential()
model.add(embedding_layer)
model.add(lstm_layer)
model.add(dropout_layer)
model.add(dense_layer)

rmsprop = RMSprop(learning_rate=0.001, rho=0.9)

model.compile(loss='categorical_crossentropy',
              optimizer=rmsprop,
              metrics=['accuracy'])
```

Kodi 11: Kompajlimi i modelit sekuencial dhe shtimi i shtresave të krijuara

Gjatë procesit të trajnimit ose të mësimit (*backpropagation*) shfrytëzohet algoritmi "*rmsprop*" me parametrat e parazgjedhur (default) [38]. Parametri "*metrics*" përpos lidhjes me trajnimin ka të bëjë edhe me pjesën e vlerësimit të performancës së modelit dhe si i tillë do të trajtohet në seksionin përkatës të punimit.

Si përfundim, mund të shfrytëzojmë funksionin "*summary*" të objektit të modelit të krijuar për të parë një tabelë përmbledhëse sikurse tek Fig. 19 të strukturës së modelit.

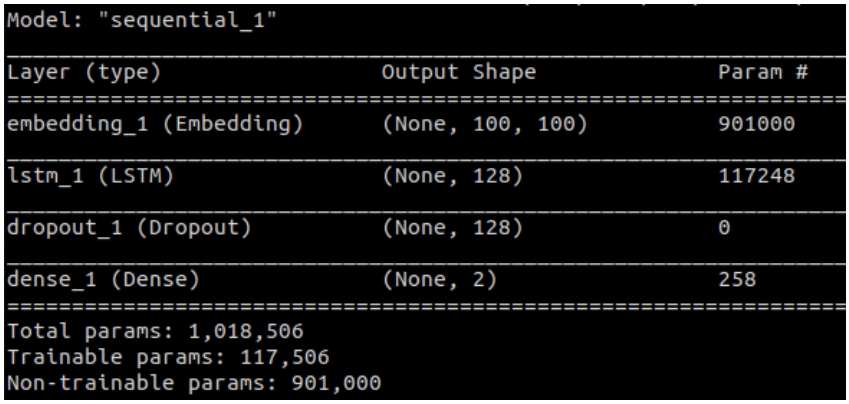

Fig. 19: Tabela përmbledhëse e strukturës së modelit të krijuar (summary)

Modeli tani është krijuar apo kompajluar (në terminologjinë e Keras) dhe është i gatshëm për tu trajnuar e pastaj për të klasifikuar.

4.3.5 Trajnimi dhe vlerësimi i modelit

Është koha që të bëhet lidhja në mes të modelit dhe të dhënave të datasetit që u përpunuan. Kjo do të realizohet gjatë procesit të mësimit apo trajnimit të modelit. Gjatë procesit të mësimit të dhënat do të ushqehen dhe do të kalojnë nëpër secilën shtresë të modelit. Algoritmi i cili do të përdoret për trajnimin e rrjetës artificiale me njësi LSTM në Machine Learning quhet *backpropagation*. Funksionimi dhe teoria matematikore që qëndron prapa *backpropagation* është jashtë fushëveprimit të kësaj teme dhe si e tillë nuk do të trajtohet detajisht. Por, është mirë të ceket se gjatë procesit të mësimit duke përdorur *backpropagation* do të bëhet kalkulimi i gabimit për secilin parametër të rrjetës dhe do të përdoret ky informacion për të rregulluar vlerat e peshave dhe bias në rrjetë. Emrin ky algoritëm e ka marrë nga fakti se pasi llogaritet një dalje, kthehet prapa për të kryer rregullimin. Zakonisht caktohet një numër i cikleve se sa shpesh do të përsëritet ky proces, kjo pasi vjen një pikë pas të cilës modeli nuk përmirësohet tutje.

Para se të paraqitet kodi se si bëhet procesi i trajnimit në program është me rëndësi kritike të trajtohen disa koncepte. Fillimisht duhet të kuptohet se procesi i *backpropagation* është proces iterativ, pra të gjitha të dhënat (i tërë dataseti) do të ushqehen disa herë. Numri i iterimeve quhet ndryshe si numri i epokave (epochs). Gjithashtu duhet cekur se gjatë një epoke, modelit nuk i prezentohen të gjitha të dhënat përnjëherë, përkundrazi, kjo bëhet tufa-tufa ose në grumbuj që quhen *batches*. Numri i iterimeve është i barabartë me numrin e tufave që duhen për të përfunduar një epokë të plotë. Nuk ka ndonjë metodë fikse për gjetjen e numrit optimal të epokave dhe të tufave prandaj në këtë punim është testuar me disa vlera të ndryshme të këtyre parametrave, dhe si përfundim 16 epoka me madhësi 32 të tufës kanë dhënë rezultate të kënaqshme. Më këto koncepte të qartësuara mund të paraqitet programi sikurse tek Kodi 12, i cili modelin e krijuar më parë e shfrytëzon për trajnim ose *fitting* në terminologjinë e Machine Learning. Funksionit "*fit*" të modelit i kalohen të dhënat për trajnim (X_train dhe y_train), të dhënat për validim ose testim (X_test dhe y_test), madhësia e tufës (batch size) dhe numri i epokave (epochs). Parametri "callback" pranon një listë të porosive (callbacks) që do të kryhen pasi modeli të ketë përfunduar fazën e trajnimit.

```
model.fit(X_train, y_train, validation_data=(X_test, y_test),
              batch_size=32
              epochs=16
              callbacks=callbacks,
              verbose=1)
```

Kodi 12: Fillimi i procesit të trajnimit (fitting) të modelit

Procesi i trajnimit do të zgjasë për një kohë, varësisht nga performanca e makinës ku zhvillohet trajnimi. Për të shpejtuar procesin mund të ngarkohet programi në ndonjë server online me pagesë ose të sigurohet harduer me performancë më të lartë, por në këtë punim duke u bazuar në madhësinë e datasetit dhe strukturën e modelit, specifikat e përmendura në 4.2 janë të mjaftueshme. Gjatë procesit të trajnimit vazhdimisht marrim mesazhe sikurse tek Fig. 20 për gjendjen e modelit dhe saktësisë aktuale (kjo për shkak se është aktivizuar opsioni *verbose* tek Kodi 12).

```
Train on 4180 samples, validate on 1394 samples
Epoch 1/16
4180/4180 [==============================] - 23s 5ms/step - los

Epoch 00001: val_accuracy improved from -inf to 0.95768, saving
Epoch 2/16
4180/4180 [==============================] - 22s 5ms/step - los

Epoch 00002: val_accuracy improved from 0.95768 to 0.97202, sav
Epoch 3/16
4180/4180 [==============================] - 25s 6ms/step - los

Epoch 00003: val_accuracy improved from 0.97202 to 0.97633, sav
Epoch 4/16
2432/4180 [================>.............] - ETA: 8s - loss: 0.
```

Fig. 20: Gjendja e modelit gjatë kalimit nëpër epokat e trajnimit

Pas përfundimit të procesit të trajnimit do të kryejmë vlerësimin e saktësisë së modelit të mësuar. Zakonisht përdoren disa parametra gjeneral për matjen e performancës sikurse në vijim:

- Accuracy – përqindja e parashikimeve të qëlluara
- Precision – përqindja e spam email që janë klasifikuar si spam dhe që realisht janë spam

Si parametër kryesor prej të cilit do të nxjerrim vlerësime për performancën e modelit do të përdorim saktësinë apo "accuracy". Për modelin që u trajnua rezulton se saktësia ka vlerën **0.982**, sigurisht kjo e vlerësuar me të dhënat për testim të ndara nga dataseti. Një vlerë e tillë konsiderohet mjaft e kënaqshme për një demonstrim bazik të konceptit. Për të arritur tek kjo vlerë janë kryer shumë cikle trajnim dhe vlerësim. Për prezentim është përzgjedhur modeli që ka treguar performancën më të mirë. Rezultatet e fituara gjatë njërit prej testeve janë paraqitur edhe grafikisht në Fig. 21:

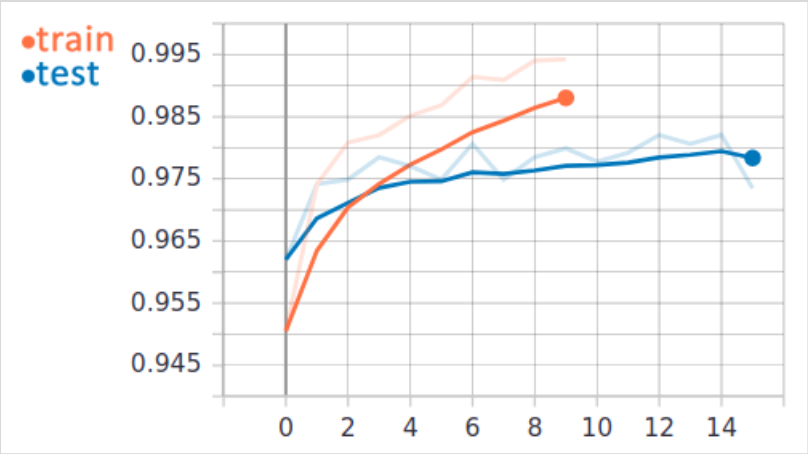

Fig. 21: Vizualizimi i saktësisë së modelit gjatë njërit prej testeve

Performanca e modelit është llogaritur duke përdorur të dhënat për testim, prandaj nuk mund të jepet garancion se modeli do të jetë në gjendje të gjeneralizojë edhe për të dhëna tjera.

4.4 Reduktimi i spam emailave përmes "proof of work"

Pasi u diskutua për mekanizmat e detektimit të spam emailave dhe si rezultat i punimit u demonstrua krijimi i një mekanizmi për kategorizim të emailave në spam ose jo spam, mbetet të trajtohet edhe tema e reduktimit të dërgimit të këtyre emailave. Sigurisht, vetë koncepti i krijimit të mekanizmave të detektimit të spam emailave është në funksion të reduktimit të vetë dukurisë. Kjo pasi nga vetë fakti se është krijuar mundësia për të detektuar një dukuri atëherë ekziston edhe mundësia që të parandalohet. Mekanizmat për detektim mund të aplikohen në anën e marrësit dhe gjithashtu në anën e pranuesit. Zakonisht, në anën e pranuesit filtrimi funksionon asisoji që emailat që pranohen dhe detektohen si spam përfundojnë në një vend të veçantë për emailat spam dhe të dyshtima. Ky vend në shumicën e rasteve quhet "junk folder". Në këtë mënyrë përdoruesit e email nuk shqetësohen nga këto lloje të emailave. Shpesh, për shkak të agresivitetit dhe gabimeve të mekanizmave detektues ndodh që një email komplet legjitim përfundon në folderin junk. Si pasojë shfrytëzuesi mund të humbasë ose mund të mos shoh emailën që i është dërguar. Prandaj do të propozohet edhe një masë tjetër shtesë (plotësuese) dhe assesi zëvendësuese për mekanizmat filtrues. Mekanizmi që do të përmendet në vijim nuk është zbulim i ri dhe sisteme të tilla janë propozuar dhe përdorur edhe më herët, prandaj edhe qëndrimi në këtë punim sa i përket kësaj metode është më shumë kritik dhe vlerësues. Mekanizmi i propozuar quhet proof of work[12] (PoW) dhe si sistem u zbulua në vitin 1993 [39]. Në hyrje të punimit u përmend se njëri prej faktorëve kryesorë që ka mundësuar popullarizimin e komunikimit me email është kostoja e ulët e dërgimit të emailave. Madje, mjafton që të ekzistojë qasja në Internet dhe dërgimi i email mund të bëhet nga shumë operatorë që ofrojnë këtë shërbim falas sikurse janë Microsoft, Google,Yahoo etj. Është pikërisht në këtë pozicion në të cilin mund të instalojmë mekanizmin proof of work. Proof of work

[12] Proof of Work mund të përkthehet në shqip si *v*ërtetësia e punës por gjatë shpjegimit do të përdoret termi origjinal në gjuhën angleze.

kërkon që për kryerjen e një veprimi siç është dërgimi i një email, të kryhet një punë nga dërguesi. Kjo është një punë e cila merr pak kohë, dhe që në fund të jetë e vërtetueshme se është kryer. Puna që do të përcaktohet të kryhet sigurisht se nuk është punë fizike, por do të jetë punë që do të kryhet nga kompjuteri i dërguesit të email. Pra, do të detyrohen resurset kompjuterike të dërguesit që të dedikojnë një kohë të caktuar nga puna e tyre për kryerjen e një pune para dërgimit të emailës. Si punë që duhet të kryhet, në këtë punim është përcaktuar llogaritja e hash[13] vlerës të një copëze të informacionit. Kjo copëz e informacionit do të quhet si header dhe do ti bashkangjitet çdo email të dërguar. Formati i saj jepet sikurse tek Kodi 13:

```
H = (receiver_email : timestamp : zero_bits : nonce : seed)
```

Kodi 13: Struktura e header-it që do ti bashkangjitet email

Kuptimi i secilit parametër është dhënë në vijim:

- **H** – headeri i emailës
- **receiver_email** – email adresa e pranuesit të emailës
- **timestamp** – koha kur është bërë kërkesa për dërgimin e emailës (në formatin unix time)
- **zero_bits** – numri i bitave zero në hash vlerën e llogaritur nga headeri
- **nonce** – një sekuencë e rëndomtë
- **seed** – vlerë e ndryshueshme që shërben për të ndikuar hash vlerën e headerit

Si hash funksion do të përcaktohet SHA256 që gjeneron hash vlera (*digest*) me gjatësi 256 bit, por në parim funksionon cilido hash funksion që gjeneron një vlerë me gjatësi të konsiderueshme dhe që është e sigurt[14]. SHA256 është një hash funksion i cili ka performancë të mirë dhe për një copëz informacioni të shkurtër si headeri që u përmend, arrin të gjenerojë hash vlerën në kohë të shkurtër. Në mënyrë që të detyrohet kompjuteri i dërguesit të emailës të kryej një punë e cila do të zgjaste për një kohë (thënë disa sekonda), do të kërkohet që hash vlera e gjeneruar të mos jetë e çfarëdoshme, por ajo t'i përmbahet një kriteri. Ky kriter do të jetë që **n** bitat e parë të vlerës së hash të header-it të jenë të barabartë me zero. Dërguesi do të ndryshojë vlerën e parametrit "seed" deri sa të gjejë një vlerë të tillë që do të gjenerojë hash vlerë me **n** bitat e kërkuar të barabartë me zero. Kjo qasje ku dërguesi duhet të provoj kombinime të ndryshme deri sa të gjejë hash vlerën që përmbushë kriterin e caktuar njihet ndryshe si *brute-force*. Në fund, headeri bashkë me vlerën e gjetur "seed" dhe me pjesën tjetër të email do të dërgohet tek marrësi. Përderisa dërguesi duhej të kryente punën e mundimshme për gjetjen e hash vlerës që përmbushë kriterin e vendosur, pranuesi mund të llogarisë vërtetësinë e punës së dërguesit pa humbur fare kohë. Për pranuesin mjafton që njëherë të llogaritet hash vlera e header-it të pranuar. Nëse ai header i përmbahet kushtit të parametrit "zero_bits", atëhere pranuesi mund të sigurohet që dërguesi ka shpenzuar kohë para

[13] Hash vlera është një vlerë numerike me gjatësi fikse që e identifikon në mënyrë unike një të dhënë. Funksioni që e mundëson këtë quhet hash funksion.
[14] Konsiderohet i sigurtë nëse nuk janë gjetur dy hyrje të ndryshme që gjenerojnë të njejtën hash vlerë, ndryshe kjo dukuri quhet hash collision.

dërgimit të emailës. Në vijim janë dhënë procedurat që duhen përcjellur nga dërguesi për të kryer punën dhe nga pranuesi për të vërtetuar kryerjen e punës:

- Dërguesi – duke ndryshuar vlerën e "seed" llogaritë hash(H) disa herë deri sa **n** bitat e parë të vlerës së hash të jenë të barabartë me zero (n = "zero_bits")
- Pranuesi – llogaritë një herë hash(H) dhe kontrollon se a përputhet numri i bitave zero me parametrin "zero_bits" të header-it të pranuar

Normalisht që parametrin "zero_bits" i cili paraqet vështirësinë e dërgimit, në një formë duhet ta diktojë marrësi sepse përndryshe dërguesit e spam emailave gjithmonë do të caktonin një numër të vogël. Numri i "zero_bits" duhet të ketë një vlerë që dërguesit t'i duhen ndoshta disa sekonda për të dërguar një email, por jo edhe kohë më e gjatë pasi kjo mund të shndërrohet në dukuri të bezdisshme për dërguesit e emailave legjitime. Gjithashtu, pranuesit i duhet të mbajë një databazë me headerët e dërguar më parë në mënyrë që dërguesit të mos nisin email të ndryshme por me një header të njejtë. Ky koncept nuk e ndalon dërgimin e spam emailave në tërësi, por ideja është që tash dërguesit e spam emailave ta kenë më të vështirë dërgimin e tufave të spam emailave si rrjedhojë e punës që duhet kryer për çdo spam që dërgojnë. Në këtë formë shpresohet se do të reduktohet numri total i spam që mund të dërgohet brenda një intervali kohor.

Një mekanizëm shumë i ngjashëm më këtë që u shpjegua është propozuar në vitin 1997 dhe quhet "Hashcash" [40]. Është evidente se koha që merr gjetja e hash vlerës përkatëse nuk do të jetë e njejtë për të gjithë. Kjo kohë varet direkt nga resurset harduerike (kryesisht fuqia procesorike) e dërguesit.

Sa më shumë resurse të ketë në dispozicion dërguesi, aq më shpejtë do të mund të dërgojë emailën. Ky sistem i propozuar, si çdo sistem tjetër, ka të metat e veta. E para është që dërguesit e emailave legjitime duhet të presin një kohë deri sa të mund të nisin e emailat e tyre. Problemi tjetër kryesorë është se dërguesit e spam emailave kanë mundësi që punën që duhet kryer ta delegojnë tek palët tjera, kryesisht përmes botnets. Problemi tjetër evident është në procesin e implementimit të këtij mekanizmi pasi për të pasur efektivitet maksimal nevojitet që të gjitha palët të marrin pjesë duke përfshirë këtu ofruesit e shërbimeve të email, implementuesit e shfletuesve të email, dhe pse jo edhe implementuesve të protokolleve për shkëmbim të emailave.

Në Fig. 22 është paraqitur grafikisht koha që duhet për të llogaritur hash vlerën që përmbushë kriteret e "zero_bits" ndërkohë që vështirësia rritet. Testimi është bërë me një implementim testues dhe të thjeshtë të algoritmit të përshkruar më lartë. Pajisja e përdorur për testim ka pasur të instaluar një procesorë Intel i7-4700HQ me shpejtësi bazë të bërthamës prej 2.4 GigaHertz. Koha e llogaritjes së hash vlerës të kërkuar, në kushte normale pëson rritje eksponenciale (edhe nëse eksperimenti nuk e reflekton këtë plotësisht).

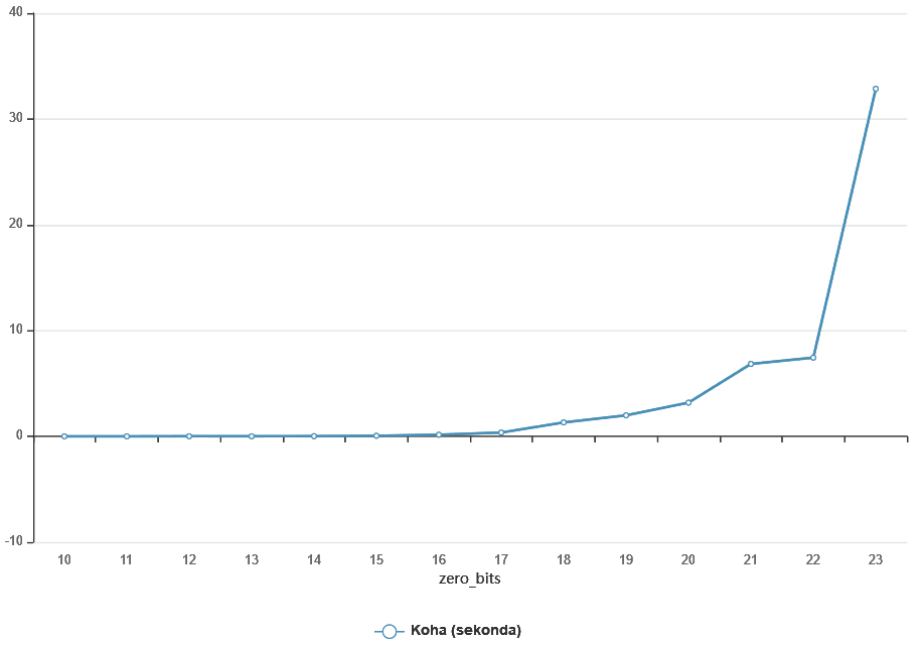

Fig. 22: Ngritja e vështirësisë (zero_bits) dhe koha (në sekonda) për të llogaritur hash vlerën

5 Diskutime dhe konkluzione

Përgjatë punimit u shpjegua dhe u argumentua rëndësia e rolit që ka posta elektronike ose email si metodë e komunikimit në kohët bashkëkohore. Si arsye kryesore të ngritjes së trendit të përdorimit të email qysh nga zbulimi u konsideruan lehtësia e përdorimit, kostoja e përdorimit dhe lehtësia në qasje. Duhet cekur patjetër se dy "kushtet" e fundit të përmendura u realizuan disa vite pas krijimit të email, kjo sepse si për shumicën e teknologjive të reja, adaptimi i tyre nga masat shkakton përmirësim dhe ulje të kostos.

Si motivim për përcaktimin e kësaj teme u prezentua dukuria e keqpërdorimit të teknologjisë email. Kjo dukuri negative ishte dërgimi i spam emailave në grumbuj. Si rrjedhojë u paraqit edhe nevoja e krijimit të mekanizmave për detektim dhe reduktim të spam emailave. Në punim u zhvillua dhe demonstrua mekanizmi për detektimin (gjegjësisht filtrimin) e spam emailave duke përdorur algoritmet e Machine Learning, konkretisht duke përdorur rrjetat neurale artificiale me njësi LSTM dhe disa koncepte të NLP siç janë word embeddings. Mekanizmi filtrues është një model i cili mundëson klasifikim binar të pjesës tekstuale të emailave në spam dhe jo spam. Pikërisht këtu është mundësia më e madhe për përmirësim.

Në versione të ardhshme mund të shqyrtohet mundësia dhe fizibiliteti i përdorimit të pjesëve tjera të email siç janë metadata, headeri, subjekti, lista e email adresave të pranuesve etj. Këto do të shërbenin si *features* shtesë për modelin dhe nëse trajtohen në mënyrë korrekte supozohet se do të përmirësohej performanca e modelit. Gjithashtu duhet të ceket se dërguesit e spam emailave viteve të fundit kanë përmirësuar jashtëzakonisht teknikat e tyre dhe emailat që i dërgojnë në ditët e sotme përmbajnë imazhe, video dhe përmbajtje tjera (HTML, CSS). Modeli i zhvilluar në punim me saktësinë e arritur është i kënaqshëm për një hulumtim dhe për demonstrim të koncepteve. Por, një mekanizëm real i cili do të përdorej për miliona përdorues do duhej që të merrte në konsideratë edhe këto përmbajtje jo-tekstuale. Edhe pse investohet shumë kohë dhe resurse, problemi i spam emailave vazhdon të mbetet njëra prej dukurive më negative dhe të rrezikshme në fushën e sigurisë së Internetit.

6 Shtesat

6.1 Lista e shkurtesave

LSTM	Long Term Short Memory
CSV	Comma-separated Values
ANN	Artificial Neural Network
ReLU	Rectified Linear Unity
NLP	Natural Language Processing
PoW	Proof of Work
SMTP	Simple Mail Transfer Protocol
HTML	Hypertext Markup Language

6.2 Lista e pjesëve të kodit

7 Referencat

[1] J. Clement, "Number of e-mail users worldwide 2017-2024, ” Statista, [Online]. Available: https://www.statista.com/statistics/255080/number-of-e-mail-users-worldwide. [Accessed March 2020].

[2] J. Clement, "Number of e-mails per day worldwide 2017-2023," Statista, [Online]. Available: https://www.statista.com/statistics/456500/daily-number-of-e-mails-worldwide. [Accessed March 2020].

[3] "What Is Spam Email?," Cisco, [Online]. Available: https://www.cisco.com/c/en/us/products/security/email-security/what-is-spam.html. [Accessed March 2020].

[4] S. Deffree, "1st spam email is sent, May 3, 1978," 3 May 2019. [Online]. Available: https://www.edn.com/1st-spam-email-is-sent-may-3-1978. [Accessed March 2020].

[5] "Email Metrics Program: Report #16 - 1st Quarter 2012 through 2nd Quarter 2014," M3AAWG, 2014.

[6] J. F. Kurose and K. W. Ross, Computer networking: a top-down approach, 6 ed., Pearson, 2013.

[7] T. Burt, "New action to disrupt world’s largest online criminal network," Microsoft, 10 March 2020. [Online]. Available: https://blogs.microsoft.com/on-the-issues/2020/03/10/necurs-botnet-cyber-crime-disrupt. [Accessed March 2020].

[8] A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2 ed., O'Reilly.

[9] "History of Machine Learning," [Online]. Available: https://www.doc.ic.ac.uk/~jce317/history-machine-learning.html. [Accessed March 2020].

[10] C. M. Bishop, Pattern Recognition and Machine Learning, Springer.

[11] R. J. Bolton and D. J. Hand, "Statistical Fraud Detection: A Review," 2002.

[12] "Artificial intelligence: Google's AlphaGo beats Go master Lee Se-dol," BBC News, 12 March 2016. [Online]. Available: https://www.bbc.com/news/technology-35785875. [Accessed March 2020].

[13] R. Amiri, H. Mehrpouyan, L. Fridman, and R. K. Mallik, "A Machine Learning Approach for Power Allocation in HetNets Considering QoS," 2018.

[14] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," 1958.

[15] A. Saravanan, "Performance of ANN in Pattern Recognition For Process Improvement Using Levenberg- Marquardt And Quasi-Newton Algorithms," 2013.

[16] F. Rosenblatt, "The Perceptron—a perceiving and recognizing automaton," *Cornell Aeronautical Laboratory*, 1957.

[17] P. Jain, "Complete Guide of Activation Functions," mc.ai, 12 June 2019. [Online]. Available: https://mc.ai/complete-guide-of-activation-functions. [Accessed March 2020].

[18] "The Intellectual Excitement of Computer Science," Stanford, 2002. [Online]. Available: https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/index.html. [Accessed March 2020].

[19] "An Introduction to Deep Learning," 20 August 2019. [Online]. Available: https://www.bouvet.no/bouvet-deler/an-introduction-to-deep-learning. [Accessed March 2020].

[20] C. Paar and J. Pelzl, "Understanding Cryptography: A Textbook for Students and Practitioners," Springer, 2019, p. 293.

[21] S. Nakov, Practical Cryptography for Developers, 2018.

[22] A. Bhowmick and S. M. Hazarika, "Machine Learning for E-mail Spam Filtering: Review,Techniques and Trends," 2016.

[23] C. Liu and S. Stamm, "Fighting unicode-obfuscated spam," Jan. 2007.," 2018.

[24] "Bayes' Theorem in email spam filtering," October 2018. [Online]. Available: http://blogs.cornell.edu/info2040/2018/10/27/bayes-theorem-in-email-spam-filtering. [Accessed March 2020].

[25] Apache, "SpamAssassin," [Online]. Available: https://spamassassin.apache.org. [Accessed March 2020].

[26] G. L. Wittel and S. F. Wu, "On Attacking Statistical Spam Filters," *CEAS*, 2004.

[27] S. H. Somanchi, "The mail you want, not the spam you don't," Gmail, 9 July 2015. [Online]. Available: https://gmail.googleblog.com/2015/07/the-mail-you-want-not-spam-you-dont.html. [Accessed March 2020].

[28] G. v. Rossum, "Comparing Python to Other Languages," [Online]. Available: https://www.python.org/doc/essays/comparisons. [Accessed March 2020].

[29] J. R. C. Nurse, "Exploring the Risks to Identity Security and Privacy in Cyberspace," *The ACM Magazine*, 2015.

[30] T. A. Almeida and J. M. Hidalgo "SMS Spam Collection v. 1," 2011. [Online]. Available: http://www.dt.fee.unicamp.br/~tiago/smsspamcollection. [Accessed March 2020].

[31] "Text Preprocessing," Keras, [Online]. Available: https://keras.io/preprocessing/text. [Accessed March 2020].

[32] A. Ferlitsch, *Regression Methods in Machine Learning: Splitting Datasets.*, Portland Data Science Group, 2017.

[33] J. Brownlee, "What Are Word Embeddings for Text?," Machine Learning Mastery, 11 October 2017. [Online]. Available: https://machinelearningmastery.com/what-are-word-embeddings.

[34] C. Perone, "Word Embeddings: A non-exhaustive introduction to Word Embeddings," April 2017. [Online]. [Accessed March 2020].

[35] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global Vectors for Word Representation," 2014. [Online]. Available: https://nlp.stanford.edu/projects/glove.

[36] J. Brownlee, "How to Reduce Overfitting With Dropout Regularization in Keras," Machine Learning Mastery, 5 December 2018. [Online]. Available: https://machinelearningmastery.com/how-to-reduce-overfitting-with-dropout-regularization-in-keras. [Accessed March 2020].

[37] Amazon AWS, "Model Fit: Underfitting vs. Overfitting," Amazon, 2020. [Online]. Available: https://docs.aws.amazon.com/machine-learning/latest/dg/model-fit-underfitting-vs-overfitting.html. [Accessed March 2020].

[38] "Optimizers," Keras Documentation, [Online]. Available: https://keras.io/optimizers. [Accessed March 2020].

[39] M. N. C. Dwork, "Pricing via Processing or Combatting Junk Mail," *CRYPTO '92*, 1992.

[40] A. Back, "Hashcash- A Denial of Service Counter-Measure," 2002.