Procjena rizika i klasifikacija sigurnosnih izvještaja u avijaciji odabranim metodama obrade teksta i tekstualne analize

Tudor, Lukrecija

Master's thesis / Diplomski rad

2021

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Science / Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:217:590147

Procjena rizika i klasifikacija sigurnosnih izvještaja u avijaciji odabranim metodama obrade teksta i tekstualne analize

Tudor, Lukrecija

Master's thesis / Diplomski rad

2021

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Science / Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:217:590147

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Lukrecija Tudor

PROCJENA RIZIKA I KLASIFIKACIJA SIGURNOSNIH IZVJEŠTAJA U AVIJACIJI ODABRANIM METODAMA OBRADE TEKSTA I TEKSTUALNE ANALIZE

Diplomski rad

Voditelj rada:
dr.sc. Tomislav Šmuc

Zagreb, 2021.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

"It seems to me that the poet has only to perceive that which others do not perceive, to look deeper than others look. And the mathematician must do the same thing."
Sofya Vasilyevna Kovalevskaya

Sadržaj

Uvod

Let avionom postao je svakodnevna pojava. Često nismo svjesni mogućih rizika i opasnosti koje let donosi. Kakvi su to rizici i kako se o njima vodi računa? Svaki eventualni rizik se opisuje, klasificira i poslije analizira. Tko je za to zadužen? 2002.godine osnovana je Agencija Europske unije za sigurnost zračnog prometa (eng. *European Aviation Safety Agency*) kraće *EASA-a*. U jednoj od njenih odredbi, između ostalog stoji:

"Treba uložiti sve napore kako bi se smanjio broj nesreća i nezgoda i osigurati visoku opću razinu sigurnosti u civilnom zrakoplovstvu.
Kako bi se poboljšala sigurnost u zrakoplovstvu, relevantne informacije o sigurnosti u civilnom zrakoplovstvu trebale bi se izvještavati, prikupljati, čuvati, štititi, razmjenjivati, širiti i analizirati, a na temelju prikupljenih podataka poduzeti odgovarajuće sigurnosne mjere." [8]

S tim ciljem uvedeno je tzv. *pojavno izvještavanje* (eng. *occurence reporting*). O svakoj nepravilnosti i svakom nepredviđenom događaju piše se izvještaj. Neki izvještaji su obavezni, neki volonterski, a mi ćemo se zadržati na obaveznim (eng. *mandatory reporting*). Cijeli popis događaja koji se prijavljuju može se naći u točki 1, članka 4 odredbe "Easy Access Rules for Occurrence Reporting"[8]
Ukratko se može reći da popis sadržava sve nepredviđene okolnosti i događaje, tehničke poteškoće, kao i ljudske ili komunikacijske pogreške. Možemo odijeliti izvještaje koji se pišu za nepravilnosti netom prije, poslije i tokom leta od onih koje se primjete pri održavanju zrakoplova. Za potrebe ovog rada korišteni su samo izvještaji o nepravilnostima prije, poslije i tokom leta. Kako piše dalje u odredbi, te izvještaje piše "glavni pilot ili, u slučajevima kada glavni pilot ne može izvijestiti o događaju, bilo koji drugi član posade koji je sljedeći u zapovjednom lancu zrakoplova".
Svi izvještaji se spremaju i ručno klasificiraju, tj. određuje im se faktor rizika prema opisanim podatcima. Ovaj rad je napravljen s ciljem da se klasifikacija napravi algoritamski, tako da se neklasificirani izvještaji sa potencijalno velikim faktorom rizika već algoritamski označe kao prioritetni. Napredniji pristup bi bio unaprijed

predvidjeti i spriječiti pojavu mogućeg događaja, prepoznati najvažnije značajke koji prethode događaju ili različite događaje sa sličnim obilježjima i možda istim uzrokom. Zračni promet se povećava iz dana u dan i informacije se gomilaju, razina sigurnosti ne može biti održana na visokoj razini ako se oslanjamo samo na ručno klasificiranje. Ono što je mana takvim izvještajima je mogućnost ljudske pogreške, kao i način na koji se izvještaj piše, nema neke ustaljene forme koja se ispuni. Također nisu svi izvještaji adekvatno napisani i događaji podrobno opisani. Pristup algoritamske klasifikacije mogao bi biti dobar motiv za uvođenje redovitih izvještaja pa da se može analizirati i svaki "ispravan" let. Mogli bi umjesto pitanja "Što je pošlo po zlu" postaviti pitanje "Zašto slična manifestacija značajki u ovom letu nije rezultirala nepravilnošću?"[7]

U ovom radu koristit će se idući matematički pojmovi:

Definicija 1. *Neka je F polje (osnovno polje, polje skalara), m, n* $\in \mathbb{N}$.
S D_{mn} *označimo kartezijev produkt* $D_{mn} = \{1, 2, ..., m\} \times \{1, 2, ..., n\}$. *Svako preslikavanje* $A : D_{mn} \to F$ *nazivamo* ***matricom tipa*** (m, n) ***nad poljem*** F. *Ako s* $\alpha_{ik} \in F$ *označimo* $A(i, k)$ *tj. vrijednost funkcije A na paru* (i, k) *onda zbog konačnosti domene matricu A zapisujemo tabelarno*

$$\begin{bmatrix} \alpha_{11} & ... & \alpha_{1n} \\ \vdots & \ddots & \vdots \\ \alpha_{m1} & ... & \alpha_{mn} \end{bmatrix} \text{ili } A = [\ \alpha_{ik}]$$

Definicija 2. ***Unitarni prostor*** *je uređeni par* (U, s) *koji se sastoji od vektorskog prostora U nad poljem* $\mathbb{R}$ *ili* $\mathbb{C}$ *i preslikavanja* $s : U \times U \to F$ *(polje) koje ima svojstva:*
1. $s(x, y) = \overline{s(x, y)}, \forall x, y \in U$ *(hermitska komutativnost)*
2.$s(x + y, z) = s(x, z) + s(y, z), \forall x, y, z \in U$ *(aditivnost na prvoj varijabli)*
3.$s(\lambda x, y) = \lambda s(x, y), \forall x, y \in U, \lambda \in F$ *(homogenost na prvoj varijabli)*
4.$x \neq 0 \implies s(x, x) > 0, \forall x \in U$ *(pozitivna definitnost)*

Definicija 3. *Neka je U unitarni prostor,* $x \in U$. *Realni broj* $\|x\| := \sqrt{\langle x|x\rangle}$ *nazivamo* ***normom*** *(modulom, duljinom)* ***vektora x***.
U slučaju $\mathbb{R}^n$ *imamo* $\|(x_1, ..., x_n)\| = \sqrt{\sum_{i=1}^{n} |x_i|^2}$

Definicija 4. *Neka su x, y nenul vektori iz unitarnog prostora U nad poljem* $\mathbb{R}$. *Tada jedinstveni broj* $\varphi \in [0, \pi]$ *sa svojstvom*

$$\cos\varphi = \frac{|\langle x|y\rangle|}{\|x\|\cdot\|y\|}$$

*nazivamo **kutom između vektora** x i y i bilježimo* $\varphi = \measuredangle(x, y)$

Definicija 5. *Neka je* Ω *neprazan prostor elementarnih događaja i* $\mathcal{F}\sigma$ *algebra skupova na njemu. Funkciju*

$$\mathcal{F} \to \mathbb{R}$$

*zovemo **vjerojatnost** na* Ω *ako zadovoljava sljedeće zahtjeve:*
A1. $P(A) \geq 0, \forall A \in \mathcal{F}$
A2.$P(\Omega) = 1$
A3.ako je dana prebrojiva familija međusobno disjunktnih skupova $(A_i, i \in I) \subseteq \mathcal{F}, I \subseteq \mathbb{N}, tj. A_i \cap A_j = \emptyset$ *čim je* $i \neq j$*, tada vrijedi:*

$$P\left(\bigcup_{i\in I}\right) = \sum_{i\in I} P(A_i)$$

*Zahtjeve A1 - A3 nazivamo **aksiomima vjerojatnosti**.*

Definicija 6. *Neka je X neprazan skup. Svaki podskup* $\rho \subseteq X \times X$ *naziva se **binarna relacija** na X. Ako je* $(x, y) \in \rho$ *pišemo* $x\rho y$.

Definicija 7. *Neka je* ρ *relacija na X. Kažemo da je* ρ ***relacija ekvivalencije** ili ekvivalencija ako ima sljedeća svojstva:*
1. $x\rho x$ *(refleksivnost)*
2. $x\rho y \implies y\rho x$ *(simetričnost)*
3. $x\rho y$ *i* $y\rho z \implies x\rho z$ *(tranzitivnost)*
Relacije ekvivalencije se često označavaju znakovima $\sim$ *(tilda),* $\simeq, \cong$ *ili* $\equiv$.

Definicija 8. *Neka je* $\sim$ *ekvivalencija na skupu X. Za proizvoljni element* $x \in X$ *uvodimo oznaku*

$$[\ x] := \{y \in X : y \sim x\}$$

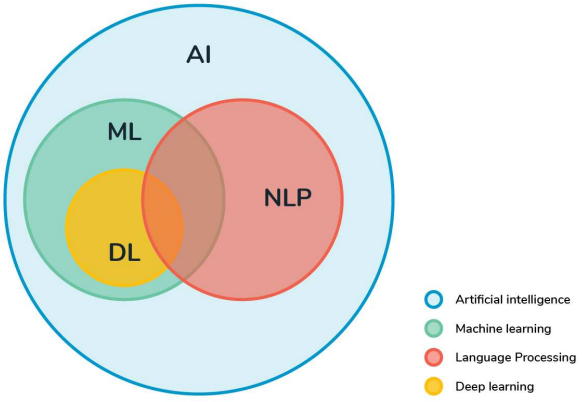
za skup svih elemenata iz X koji su ekvivalentni s x.
Skup $[\ x]$ *ćemo nazivati **klasom ekvivalencije** reprezentiranom elementom x.*

Poglavlje 1

Strojno učenje i NLP

Umjetna inteligencija (eng. *Artificial Intelligence*, kraće *AI*) sposobnost je digitalnog računala ili računalom upravljanog robota da izvršava zadatke uglavnom povezane s inteligentnim bićima. Pojam se često odnosi na razvoj sustava s ciljem procesuiranja intelektualnih procesa, poput sposobnosti rasuđivanja, otkrivanja značenja, generaliziranja ili učenja iz prošlih iskustava.[6]
Strojno učenje, (eng. *Machine learning*) definiramo kao skup metoda *umjetne inteligencije* koje automatski otkrivaju uzorke u danima podacima, a zatim koriste otkrivene uzorke za predviđanje budućih podataka ili za samostalno donošenje odluka neke druge vrste. Strojno učenje obično se dijeli na dvije glavne vrste, nadgledano i nenadgledano učenje (eng. *supervised and unsupervised learning*). U nadgledanom pristupu učenju, cilj algoritma je preko dobivenog skupa podataka (u obliku uređenih parova) $D = \{(x_i, y_i) : i = 1, .., N\}$ (pri čemu D označava skup podataka za treniranje modela, a N broj elemenata tog skupa), što bolje replicirati funkciju f kojoj su ulazne vrijednosti x_i a izlazne vrijednosti $f(x) = y_i$ te tu funkciju primijeniti na novi skup podataka $X = \{x_i : i = 1, .., M\}$ u cilju dobivanja skupa $F = \{f(x_i) : i = 1, .., M\}$, gdje M označava broj elemenata testnog skupa X (skup na kojem se istrenirani model evaluira).[11] (Repliciranje funkcije ne postiže se nikada potpuno nego algoritam svakom iteracijom sve bolje aproksimira ishod preslikavanja, mogli bismo figurativno reći da algoritam "uči" funkciju).
Klasifikacija spada u probleme nadgledanog učenja. Cilj klasifikacije je "naučiti" preslikavanje od ulaza X na izlaze y, gdje je $y \in \{1, ..., C\}$, pri čemu je C broj klasa kojima ulazni podaci pripadaju. U slučaju $C = 2$, to se naziva *binarna klasifikacija* (u tom slučaju često pretpostavljamo $y \in \{0, 1\}$), a ako je $C > 2$, naziva se *višeklasnom klasifikacijom*. Tzv. naučeno preslikavanje primjenjuje se na skup podataka kojim ćemo model evaluirati. U krajnjem slučaju, ako je model zadovoljio kriterije, primijeniti ćemo ga na podacima koje želimo računalno klasificirati (dodijeliti

svakom podatku jednu od *C* klasa za koju je algoritam procijenio da joj podatak priprada).
Dubinsko učenje (eng.*Deep learning*) je podvrsta *strojnog učenja* nadahnuta strukturom ljudskog mozga. Pripadajući algoritmi ustraju u donošenju sličnih zaključaka kao i ljudi, s naglaskom na kontinuiranu analizu podataka. Da bi to uspjelo, dubinsko učenje koristi zadanu višeslojnu strukturu algoritma koja podsjeća na ljudski mozak, a zove se *Neuronska mreža* (eng. *Neural networks*).[12]
NLP (eng. *Natural language processing*) je grana *umjetne inteligencije* koja proučava i kreira interakcije između računala i ljudskih (prirodnih) jezika. Sadržana je od niza metoda obrade jezika nastalih iz računalne lingvistike a podrazumijeva metode iz različitih disciplina (računalstva, lingvistike, podatkovne znanosti i sl.) a sve u svrhu što boljeg razumijevanja ljudskog jezika u pisanom i verbalnom obliku.
NLU (eng. *Natural language understanding*) obuhvaća jedan dio *NLP-a* koji koristi analizu sintakse i semantičku analizu teksta i govora kako bi se približili kontekstualnom značenju rečenice. Sintaksa se odnosi na gramatičku strukturu rečenice, dok semantika aludira na njenu namjeru.
NLG (eng. *Natural language generation*) još je jedan dio *NLP-a*. Za razliku od *NLU-a* koji se fokusira na računarsko "čitanje s razumijevanjem", *NLG* računalu omogućuje pisanje u smislu stvaranja odgovora na ljudskom jeziku temeljem unesenih podataka. (Naravno, takav se tekst može pretvoriti u druge formate kao što je npr. govor) [9]



Slika 1.1: Dijagram strukture *umjetne inteligencije*, preuzeto sa [4]

1.1 Opis problema

Metodama NLP-a i strojnog učenja dobiveni podaci za rad trenirani su za klasifikaciju. Da bi algoritam kao ulazne informacije koristio tekst prvo ga mora nekako preformulirati u jezik koji računalo razumije, a to su brojevi. Postoje razvijene metode brojčane reprezentacije teksta, ali o tome detaljnije nešto kasnije. U ovom radu korištene su dvije metode reprezentacije teksta (TF-IDF i Bert Encoding) i dva klasifikacijska modela (Random Forest i BERT). Osovni problem na koji se nailazi je nebalansiranost podataka s obzriom na klase što se očituje i na rezultatima. U daljnjim poglavljima detaljno su opisane metode reprezentacije teksta, klasifikacijski modeli korišteni za dobivanje rezultata kao i metode evaluacije korištenih modela.

Poglavlje 2

Skup podataka

2.1 Podaci

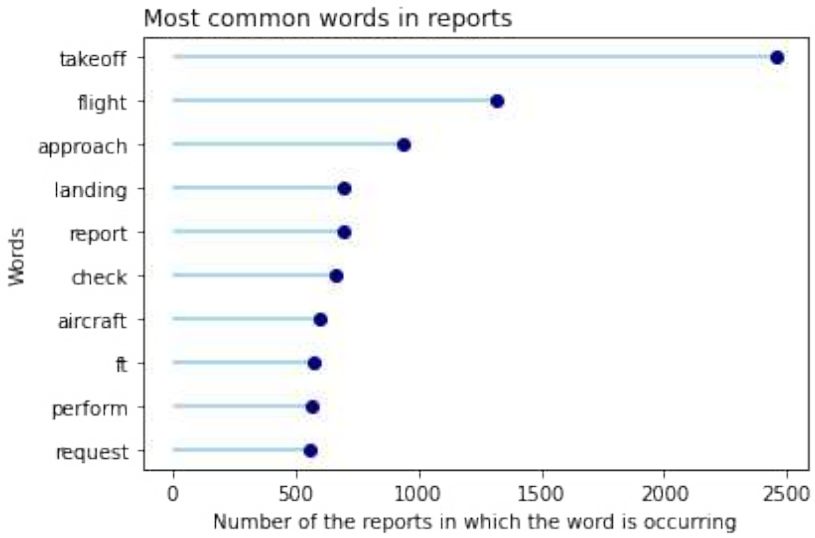
Skup podataka sastoji se od 3633 sigurnosna izvještaja aviokompanije *Adria Airways* napisana u slučaju bilo koje nepravilnosti pri polijetanju, slijetanju ili tokom samog leta. Osim izvještaja pamte se informacije o lokaciji događaja, kratki naziv nastalog problema te za 3079 izvještaja ocjena i faktor rizika od nastale nepravilnosti.
Faktor rizika s obzirom na svaki pojedini izvještaj dodijeljen je od strane stručne osobe prema odredbi EASA-e. Dodijeljeni faktori su u neekvidinstantnoj skali od 1 do 2500. Grupacijom izvještaja prema faktoru rizika dodjeljuje se ocjena rizika 0, 1 ili 2.
Na tablici prikazanoj na slici 2.1 grupacija ocjene prema faktoru rizika označena je razlikom u boji. Zelena polja označavaju faktore rizika koji spadaju u klasu 0, žuta polja označavaju faktore koji spadaju u klasu 1, a crvena polja u klasu 2.

	EFFECTIVE Remaining Barriers	LIMITED EFFECTIVENESS of Remaining Barriers	MINIMAL EFFECTIVENESS of Remaining Barriers	NOT EFFECTIVE Remaining Barriers
CATASTROPHIC ACCIDENT Loss of aircraft or multiple fatalities (3 or more)	50	102	502	2500
MAJOR ACCIDENT 1-2 fatalities, multiple serious injuries, major damage to the aircraft	10	21	101	500
MINOR INJURIES OR DAMAGE Minor injuries, minor damage to aircraft	2	4	20	100
NO ACCIDENT OUTCOME No potential damage or injury could occur	1	1	1	1

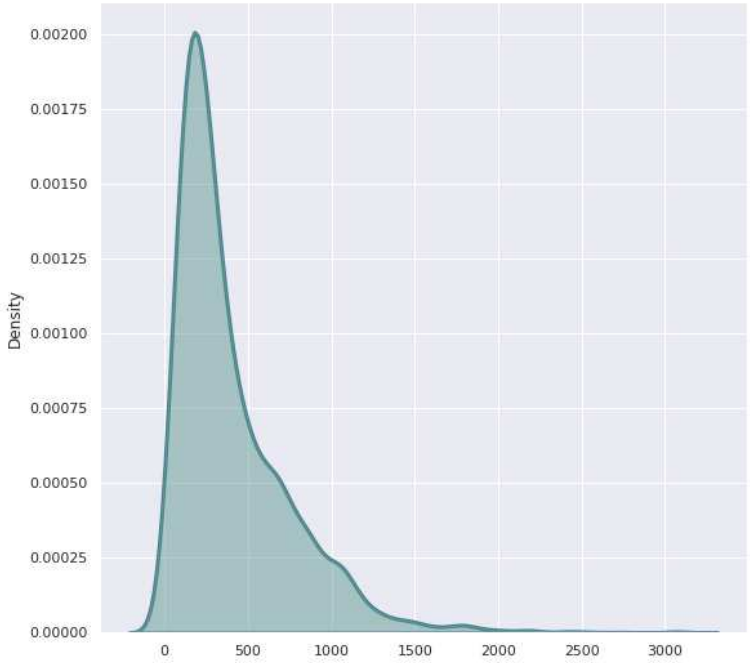
Slika 2.1: Safety Event (Occurrence) Risk Assessment Matrix Details

S obzirom da je "crvenih" izvještaja, tj onih s ocjenom 2, jako malo (svega 3 izvještaja na cijelom skupu podataka) očita je potreba za novom raspodjelom. Kako neekvidistantna skala rizika od događanja mogućih nepravilnosti krajnjem korisniku nije toliko korisna koliko predviđanje posljedica sličnih nepravilnosti u budućim letovima, nova raspodjela je kreirana također prema tablici na slici 2.1 ali s obzirom na predviđene ishode ovisno o njihovom faktoru rizika.
Takva podjela sadrži 3 klase i dijeli podatke na događaje bez rizika od nesreće u ishodu, događaje s rizikom za manje ozljede ili štete i događaje sa rizikom od znatnih ili katastrofalnih nesreća u ishodu. Zadnja klasa sastoji se od dvije spojene klase, vidljive na slici 2.1 i to iz dva razloga. Prvi je brojčani manjak događaja sa rizikom za nesreće u ishodu, a drugi razlog je što samo uključivanje većih šteta i ljudskih žrtava okarakterizira svaku nepravilnost kao potencijalno fatalnu i krajnji cilj je unaprijed prepoznati sve takve nepravilnosti.

Slika 2.2: Broj izvještaja po klasama



Slika 2.3: 10 najučestalijih riječi po izvješćima



Slika 2.4: Histogram broja riječi po izvještaju

2.2 Transformacija tekstualnih podataka

Tekst, za razliku od numeričkih podataka, nije moguće direktno analizirati. Rijetko kada ima smisla gledati samo pojedinačne znakove. Bez da ih posložimo redoslijedom u cjelinu ne možemo dobiti predodžbu o tekstu. Većina algoritama za transformaciju teksta radi po principu preoblikovanja teksta u vektorski oblik. Takve transformacije zovemo i reprezentacije tekstualne forme. Prvotni algoritmi za transformaciju teksta to rade tako što svakoj jedinstvenoj riječi dodjele numeričku vrijednost, tako da se cijeli tekst može predočiti nizom brojeva. Ovakvom osnovnom transformacijom brojevi samo zamjenjuju riječi, ali ne doprinose razumijevanju te riječi. Na osnovu dobijenog niza nije moguće dobiti kontekstualno značenje za sve riječi koje se nalaze u korpusu, ali analizom frekvencije pojave riječi u tekstu potencijalno možemo dobiti koje su riječi značajnije, a koje manje značajne u tom tekstu.

2.3 TF-IDF reprezentacija

Sada možemo predstaviti tzv. *BOW* ('*Bag Of Words*') model reprezentacije. *BOW* reprezentacija svakoj riječi u tekstu dodjeli broj koji označava koliko se puta ta riječ pojavila u tekstu. Npr. rečenica "*Dok Lukrecija voli čitati triler knjige, Elio uživa gledajući triler filmove*" prezentirala bi se sa:

Dok: 1, *Lukrecija:* 1, *voli:* 1, *čitati:* 1, *triler:* 2, *knjige:* 1
Elio : 1, *uživa:* 1, *gledajući:* 1, *filmove:* 1

Kada želimo BOW metodom uspoređivati više tekstova napravit ćemo dokument-pojam matricu. U kojoj svaki redak matrice označava reprezentaciju jednog dokumenta s obzirom na sve različite riječi koje se u svim tekstovima pojavljuju. Te riječi se nazivaju tokenima i svaki stupac reprezentira pojavu jednog tokena kroz sve dokumente. Vrijednosti unutar matrice bit će, kao gore, izračunate vrijednosti frekvencija riječi u tekstu.
U ovom radu je korištena modifikacija *BOW* reprezentacije koja se u primjeni pokazala uspješnom iako kao ni BOW ne sačuva kontekst riječi. Ta modifikacija je poznata pod nazivom TF-IDF reprezentacija (*Term Frequency-Inverse Document Frequency-Formula*) i od *BOW* reprezentacije se razlikuje po tome što se u matrici na drugačiji način računaju vrijednosti. Vrijednost svakog elementa dokument-pojam matrice računa se po formuli:

$$tfidf(w,d,D) = tf(w,d) * idf(w,D)$$

gdje vrijedi:

$$tf(w,d) = log(1 + f(w,d))$$
$$idf(w,D) = log(\frac{N}{f(w,D)})$$

pri čemu je N veličina skupa podataka tj. ukupan broj tekstova koje proučavamo, $f(w,d)$ je frekvencija tokena w u dokumentu d, a $f(w,D)$ broj dokumenata (maksimalno N) u kojima se pojavljuje token w.
Ovim pristupom bolje se prezentira značaj same riječi. Što je riječ učestalija u skupu podataka to je broj $\frac{N}{f(w,D)}$ manji. Zamislimo da se neka riječ pojavljuje u 95% izvještaja. Tada je logično zaključiti da ona nije od velikog značaja za razlikovanje izvještaja s obzirom na klase, kao i situacija kada bi se neka riječ pojavljivala u samo jednom izvještaju. Od većeg značaja su riječi koje se na primjer pojavljuju u 30% izvještaja. Primjer vrsta riječi koje se često pojavljuju i prividno ostavljaju dojam velikog značaja su npr. veznici. Oni nam zapravo o samom sadržaju ne govore ništa. Takve riječi mogu se čak i izbaciti iz korpusa. Slično je i s riječima koje se pojavljuju u manje izvještaja od neke, unaprijed dogovorene, donje granice.

2.3.1 Reprezentacija za ML modele

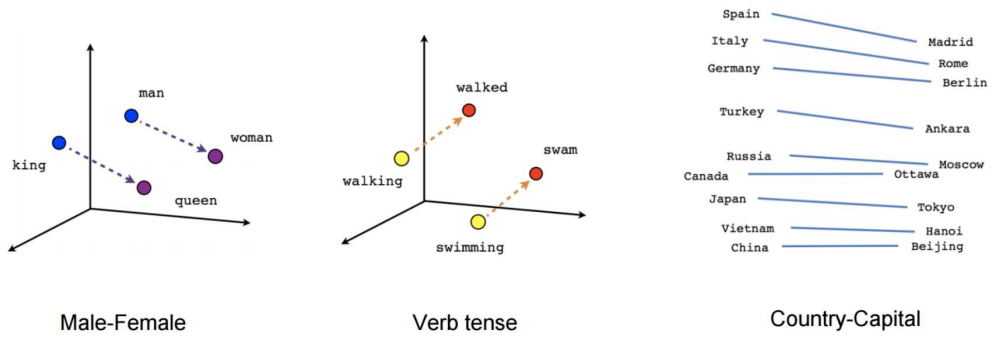
TF-IDF reprezentacija s jedne strane dobro opisuje strukturu tekstova, ali nema sposobnost da uhvati kontekstualno značenje. Vrijeme je da predstavimo modernije metode.
Word2vec je model kreiran i patentiran od tima Google-ovih istraživača na čelu s Tomasom Mikolovim, a objavljen je 2013.godine. Ono što je bio cilj *Word2vec* modela je također riječi prikazati nizom brojeva, no u ovom slučaju, vektorom (unaprijed zadane dužine, uobičajeno između 100 i 1000) reprezentirati riječi tako da se pokušaju zadržati njihovi međusobni odnosi. Npr. izračunata sličnost između dva vektora koji reprezentiraju riječi *Berlin* i *Njemačka* ista je kao sličnost između vektorskih reprezentacija riječi *Rim* i *Italija*.[3]
Za sličnost vektora odabire se mjera kosinus sličnosti, izračunata na ovaj način:

$$Cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \, \|\vec{b}\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2} \sqrt{\sum_1^n b_i^2}}$$

Slika 2.5: a i b su vektori duljine n koji prikazuju 2 različite riječi

Reprezentacije i modeli koji će se koristiti u ovom radu, uvelike se oslanjaju na

word2vec reprezentaciju. S obzirom da je za razumijevanje konteksta potrebno jako puno prijašnjeg iskustva u kojima se riječi možda ne koriste doslovnom pojmovnom značenju, modeli reprezentacije se unaprijed treniraju (eng. *pretrained*) na jako velikim bazama tekstova kao što je *Wikipedia* i slično. Na temelju tih bazi stvara se rječnik vektorskih prikaza pojmova bazirajući se na njihovu međusobnu sličnost. Takvi prikazi riječi biti će ulazne informacije za modele koji će prema njima dalje računati sličnost i zavisnost riječi u tekstovima pa kodirati ulazni vektor kroz više slojeva u želji da se kroz svaki sloj brojčane vrijednosti što bolje usklade jedne s drugima i daju vjerniju aproksimaciju teksta. Važno je napomenuti da je, iako je tekstualna reprezentacija od 2013. dosta napredovala, *Word2vec* model bio je svojevrsna revolucija u računalnom "čitanju" teksta jer je dotad bio jedini model koji se približio ideji razumijevanja konteksta.



Slika 2.6: prikaz odnosa ovih vektorskih reprezentacija čest je primjer promjene koju je uveo *word2vec*

2.3.2 Priprema podataka

Podaci koji su dani na korištenje su grubi, nefiltrirani, neuređeni podaci. Ovisno o reprezentacijama koje će se koristiti, potreba za obradom dokumenata se razlikuje. Prije primjene bilo koje metode potrebno je podatke obraditi i dovesti u oblik koji pridonosi boljim i smislenijim rezultatima. Ta obrada je proces od više koraka.
Prvi korak su tzv. NA ('Not Available') vrijednosti. Svi oni dokumenti kojima izostaje tekst izvještaja se izbacuju. Također, svi izvještaji koji nisu (s velikom vjerojatnošću) engleskog govornog područja se izbacuju.

S tako filtriranim skupom podataka koji imaju ocjenu rizika ostalo je 2851 izvještaj spreman za daljnju obradu.
S obzirom da izvještaje pišu sami piloti, stil pisanja kao ni izražavanje nisu ujednačeni. U to spada i knjiga kratica u zrakoplovstvu koja se, zbog ljudskog faktora, nekada i ne koristi baš precizno.
Prolaskom kroz konkretne izvještaje i knjigu kratica sastavljena je lista najčešće korištenih i sve su usuglašene. Neke su sačuvane u obliku kratice, a neke u punoj tekstualnoj verziji, ovisno o autorovoj slobodnoj procjeni boljeg kontekstualnog značenja. Na slici 2.7 je popis svih kratica i njihovih zamjena. Što se tiče *BERT* modela ovo su jedine transformacije koje su napravljene.

Idući korak je uređivanje samih izvještaja za TF-IDF reprezentaciju. Iz svakog izvještaja izbacimo sve interpunkcijske znakove, simbole i brojeve. Također je potrebno sva velika slova zamijeniti malima, ili obrnuto, jer želimo izbjeći situaciju u kojoj se npr. riječi 'tower' i 'Tower' tretiraju različito. Neće nam biti važno koja se riječ gdje nalazi tako da ćemo riječi s početka rečenice tretirati jednako kao i one u sredini i na kraju. Također kako ne možemo "uhvatiti" kontekst izvještaja nije nam važno jesu li rečenice upitne ili izjavne pa nam interpunkcija malo znači.
Inače, važan korak u procesuiranju tekstova je filtriranje nepotrebnih elemenata, a najbitnije filtriranje je izbacivanje tzv. stop riječi.
Jedan primjer takvih riječi su već spomenuti veznici. Na primjer, riječi kao *'a', 'an', 'the', 'and', 'but', 'if', 'or', 'as', 'until', 'while', 'me', 'you', 'he', 'his', 'her'*... rade veći šum u skupu postojećih riječi nego što doprinose značenju. Iz tog razloga, za određene reprezentacije takve se riječi izbacuju. Cijeli popis stop riječi engleskog jezika koje se koriste u ovom radu preuzet je iz open source biblioteke ***spaCy***.
Dosadašnji postupci nam nažalost rješavaju samo dio problema. Drugi veliki problem kod ovakve reprezentacije teksta je činjenica da bi riječi koje su u množini ili u različitim padežima i vremenima, bile prezentirane različitim brojevima i samim time "pročitane" kao u potpunosti različite riječi. Primjera radi, uzmimo riječi 'run' i 'running'. Bespogovorno te dvije riječi nisu identične, ali značenjski se minimalno, zanemarivo razlikuju. Međutim, moguće je da se tokom procesa reprezentacije, jer morfološki nisu iste, te dvije riječi tretiraju kao različite. Naš cilj je jako slične riječi po značenju tretirati kao iste. U tu svrhu se radi *stemanje* i/ili *lematizacija*. Oba navedena procesa svaku riječ koja se pojavljuje u tekstu preoblikuju tako da sve jako slične riječi dobiju jednak oblik.
Lematizacija je grubo rečeno odbijanje prefiksa i sufiksa riječi dok bi stemanje (eng. *stemming* izvedenica od riječi *stem* što znači *stabljika*) mogli objasniti sa 'svođenje na korijen' jer je to, između ostalog, izraz za odvajanje stabljika od kori-

rwy	runway
r/w	runway
rwy	runway
ry	runway
rnwy	runway
ra	resolutionadvisory
told	takeoff and landing
to	takeoff
t/o	takeoff
take-off	takeoff
toga	takeoff go around
to/ga	takeoff go around
tof	time of flight
agl	abovegroundlevel
above ground level	abovegroundlevel
gs	groundspeed
g/s	groundspeed
ground speed	groundspeed
a/p	autopilot
auto-pilot	autopilot
ap	autopilot
g/p	guidance panel
a/c	aircraft
ac	aircraft
acft	aircraft
aeroplane	aircraft
aoa	angleofattack
angle of attack	angleofattack
air traffic control	atc
dow/doi	dowdoi
dg	dangerousgoods
dgr	dangerousgoods
dgor	dangerousgoods

Slika 2.7: Popis svih kratica i njihovih supstitucija korištenih u ovom radu

jena biljke. Ovim procesima se dobija puno bolja prezetacija teksta međutim i dalje ostaje problem konteksta, žargona ili sinonima. Na primjer pogledajmo rečenice: *"I have just booked a flight for tomorrow"* i *"Can you lend me that book you were talking about?"* Riječi "booked" i "book" će biti svedene na svoj osnovni oblik - "book", međutim imenica "book" i glagol "to book" ne znače isto. Kontekstualno se razumije o čemu je riječ, međutim računalo te dvije riječi poveže kao da su istog značenja. Rješenje ovih problema zahtjeva kompleksnije algoritme.

Prije samih primjera stemanja i lematizacije na konkretnim riječima, važno je uočiti jednu razliku između ta dva procesa. Stemanjem se riječ poistovjećuje sa svojim korjenom i preoblikuje u njega te zato često nema neko konkretno značenje sama za sebe, dok lematizacija svaku riječ preoblikuje u njenu baznu formu uzimajući u obzir da izmijenjena riječ sama ima značenje. U ovom radu je za obradu teksta korištena lematizacija te je za nju također korištena open source biblioteka ***spaCy***[15].

```
'During climb at around FL200 we noticed that left engine oil temp
erature was rising quite quickly. When indication went into a red
range I reduced left thrust to idle and ordered QRH for high oil t
emperature. I declared PAN PAN to Vienna Radar and turned around t
o DOL and we started descent. QRH was completed and left engine wa
s left at idle accordingly. ( the highest peak was 176 C ). Oil pr
essure and oil quantity was within normal range. During descent oi
l temperature was slowly dropping and reached normal range  ( 95 C
) at around DOL. We decided to land with normal thrust setting on
both engines and we carefully monitored oil temperature. We also c
hecked QRH for Single Engine Approach and Landing and we were brie
fed accordingly,\r\nAfter landing we declared Unfit to fly.'

'['climb', 'notice', 'leave', 'engine', 'oil', 'temperature', 'ris
e', 'quickly', 'indication', 'go', 'red', 'range', 'reduce', 'leav
e', 'thrust', 'takeoff', 'idle', 'order', 'qrh', 'high', 'oil', 't
emperature', 'declare', 'pan', 'pan', 'takeoff', 'vienna', 'radar'
, 'turn', 'takeoff', 'dol', 'start', 'descent', 'qrh', 'complete',
'leave', 'engine', 'leave', 'idle', 'accordingly', 'high', 'peak',
'c', 'oil', 'pressure', 'oil', 'quantity', 'normal', 'range', 'des
cent', 'oil', 'temperature', 'slowly', 'drop', 'reach', 'normal',
'range', 'c', 'dol', 'decide', 'takeoff', 'land', 'normal', 'thrus
t', 'set', 'engine', 'carefully', 'monitor', 'oil', 'temperature',
'check', 'qrh', 'single', 'engine', 'approach', 'landing', 'brief'
, 'accordingly', 'land', 'declare', 'unfit', 'takeoff', 'fly']'
```

Slika 2.8: Primjer izvještaja (ID = 2528715) prije i nakon obrade

Poglavlje 3

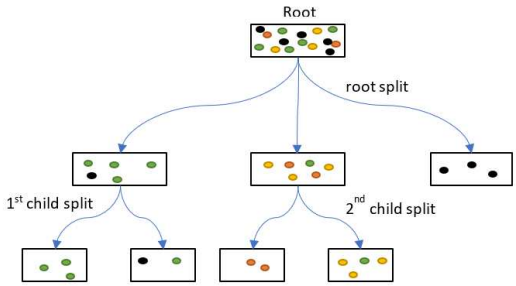
Klasifikacijski modeli

3.1 Slučajna šuma (eng. *Random Forest*)

Algoritam slučajnih šuma (eng. *random forest*) jedan je od algoritama iz područja strojnog učenja (eng. *machine learning*) koji se koristi za raspoznavanje uzoraka i regresijsku analizu. Autor ovog algoritma je profesor Leo Breiman sa američkog sveučilišta Berkely, a glavna značajka algoritma je da on pri raspoznavanju ne koristi samo jedan klasifikator nego kreira veći broj klasifikatora u obliku stabala odluke (eng. *decision tree*) od kojih svako sudjeluje u izglasavanju konačnog rezultata.

Definicija 9 (Slučajna šuma)**.** *Slučajna šuma je klasifikator koji se sastoji od skupa stablasto strukturiranih manjih klasifikatora* $\{h(x, \theta_k), k = 1, ...\}$ *gdje su* $\{\theta_k\}$ *nezavisni, jednako distribuirani slučajni vektori. Svako stablo odluke proizvodi svoj klasifikacijski rezultat s obzirom na ulazni vektor x. Konačan rezultat metode je najčešće predviđena klasa među svim stablima odluke.[5]*

Metoda slučajne šume, pripada jednoj od popularnijih tzv. *ansambl metoda*. U našem slučaju, *slučajna šuma* je (kao gore u definiciji) ansambl više individualnih *stabala odluke*.
Stablo odluke sastoji se od čvorova, grana i listova (vidljivo na slici 3.1). Čvor u vrhu zove se još i *korijen* (eng. *root*) i predstavlja osnovnu značajku prema kojoj se podaci granaju u nove čvorove. Cilj je u krajnjim čvorovima koje zovemo *listovi* doći do odgovora koju klasu ćemo dodijeliti ulaznim podatcima. Na slici 3.1 ulazni skup podataka je skup točaka različite boje. Boju možemo shvatiti kao značajku klasificiranja u ovom slučaju. U svakom čvoru dolazi do razdvajanja ulaznog skupa na dva ili više čvorova i tako sve do listova. Cilj je da u svakom listu prevlada pripadnost jednoj klasi. Ono što nije banalno odrediti je način na koji se ulazni podaci



Slika 3.1: Općeniti primjer jednog stabla odluke

u svakom čvoru razdjeljuju kao i odrediti značajke po kojima se dijeli ulazne podatke u pojedinom čvoru. Te odluke se donose pri izgradnji stabla odluke na skupu za treniranje modela. (Primjer na slici je pojednostavljen pa je na njemu prikazano stablo odluke sa podacima određenima samo jednom značajkom - bojom, ali često u primjeni značajki ima puno više i neke su važnije za klasifikaciju od drugih, neke su značajnije samo za jednu klasu a za druge nisu i sl.)
Jedan od načina za izgradnju je krenuti od prvog čvora, korijena i za njega odlučiti po kojoj značajki podijeliti ulazni skup. Kada ga podijelimo na nove čvorove ponavljamo postupak sve dok ne dođemo do listova stabla. Broj koraka i razina grananja može se odrediti na dva načina. Jedan način je da se fiksan broj koraka odredi unaprijed, a drugi način je da se zada određena mjera efikasnosti i njena vrijednost koja mora biti zadovoljena. Kada se zadovolji ta vrijednost čvor se više ne grana i ne radi se nova podjela iz njega.[18]
Opišimo malo pobliže odabir značajki i podjelu u svakom čvoru. Ona se može napraviti tako da se isproba podjela po svim značajkama i odredi koja značajka je bila najefikasnija u podjeli, pri čemu efikasnost mjerimo različitim načinima od kojih je najjednostavnija već spomenuta *točnost* (eng. *accuracy*). Međutim u ovom radu je za efikasnost značajki stabala odluke korišten *Ginijev indeks nečistoće* (eng. *Gini impurity*). Koncept ovakve mjere je da poprima vriijednosti proporcionalno raznolikosti skupa tj. ako je skup 100% homogen (sadrži samo pripadnike jedne klase) koeficijent je 0, a što se raznolikost pripadajućih klasa elemenata skupa povećava, to se povećava i Ginijev indeks. Ginijev indeks u svakom čvoru računa se po formuli:

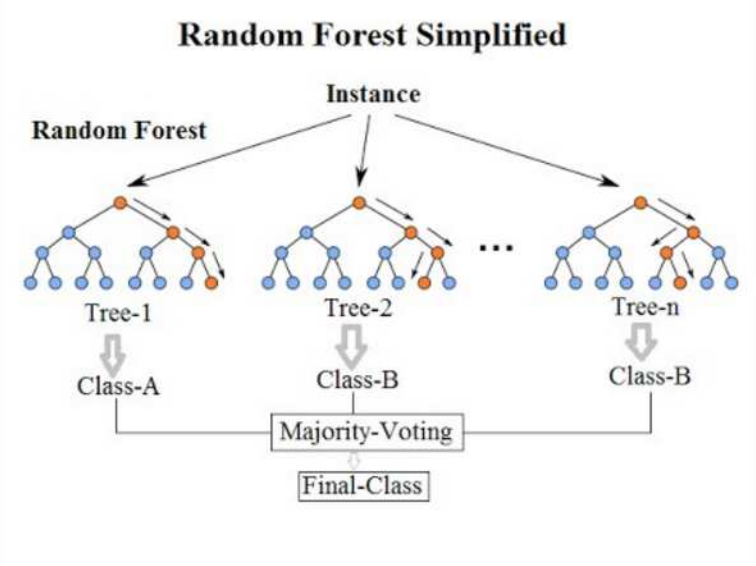
$$\sum_i w_i \times p(i) \times (1 - p(i))$$

pri čemu i označava i-tu klasu, $p(i)$ vjerojatnost da ćemo nasumičnim odabirom s

mogućnosti ponavljanja (iz skupa podataka u čvoru) dobiti element klase i, a w_i težinu (koeficijent važnosti) klase i. Ukoliko želimo da svaka klasa "teži" jednako, odrediti ćemo $w_i = 1$, $\forall i$. Međutim, ako mjerimo Ginijev indeks neke podjele, onda ćemo izmjeriti Ginijev indeks početnog čvora i aritmetičku sredinu Ginijevih indeksa svih čvorova koji nastaju podjelom iz početnog čvora. Tada bi nam težine mogle biti od koristi. Da pojasnimo pretpostavimo sljedeću situaciju: Početni čvor ima skup od 100 elemenata. Podjelom, nastaju dva skupa, jedan čvor sadrži 99 elemenata, a drugi samo 1. U tom slučaju će Ginijev indeks drugog skupa biti 0 jer je skup u potpunosti homogen, međutim ta homogenost jednočlanih skupova nam znači malo s obzirom da u drugom čvoru imamo 99 elemenata za razdijeliti, dakle nismo se odmakli puno dalje od početnog problema, a aritmetička će sredina Ginijevih indeksa nastalih skupova biti najviše 50%, dok je zapravo možda puno veća.[14] U situacijama kao što je ova korisno je imati težine svakog nastalog skupa. Ona se odredi omjerom elemenata nastalog i početnog skupa, tako bi u gore navedenoj situaciji w_1 bila $\frac{99}{100}$ a w_2 $\frac{1}{100}$.

Sada kada smo pobliže objasnili princip klasifikacije stabla odluke, možemo pojasniti kako funkcionira algoritam slučajne šume. Prilikom treniranja, algoritam slučajne šume stvara velik broj takvih stabala. Kasnije, kada se izgrade sva stabla i istreniraju podaci, dolazi na red testiranje modela tj. klasifikacija podataka iz testnog skupa. Za klasifikaciju svakog ulaznog podatka, sva stabla unutar algoritma (već izgrađena na podacima za treniranje modela) daju glas jednoj od klasa za koju predviđaju da joj ulazni podatak pripada. Konačna izlazna vrijednost algoritma (cijele šume) je ona klasa koju je predvidio najveći broj stabala.[10]

Ali, ako je stablo odluke izgrađeno najefikasnije moguće, po čemu će se onda stabla unutar šume razlikovati u strukturi i konačno u predviđanju? Važno je napomenuti da stablo odluke spada u tzv. *pohlepne algoritme* (eng. greedy). Pohlepnima se okarakteriziraju oni algoritmi koji u svakom koraku traže optimalno rješenje u tom trenutku, pritom možda generalno skup "najboljih" koraka ne daje optimalno konačno rješenje. I zato se svako od spomenutih stabala trenira na različitom poduzorku koji je jednake duljine kao što je i originalni set za treniranje (osim ako se ne naglasi drugačije) ali poduzorak za pojedino stablo dobiven je uzorkovanjem s ponavljanjem (iz originalnog seta), pa postoji mogućnost višestrukog izbora nekih elemenata u istom skupu za izgradnju. Na primjer, zamislimo da je naš skup podataka za treniranje $[1, 2, 3, 4, 5, 6]$, tada bi uzorak dan jednom od stabala odluke mogao biti $[1, 2, 2, 3, 4, 4]$. Oba skupa su duljine 6, ali zbog dozvoljenog ponavljanja u uzorkovanju, elementi 2 i 4 se pojavljuju 2 puta dok elementi 5 i 6 izostaju. U nekom drugom stablu odluke moglo bi se dogoditi obrnuto. Zbog činjenice da su uzorci odabrani slučajnom metodom, međusobno su nekorelirani što je potreban preduvjet za dobar rezultat metode slučajne šume.



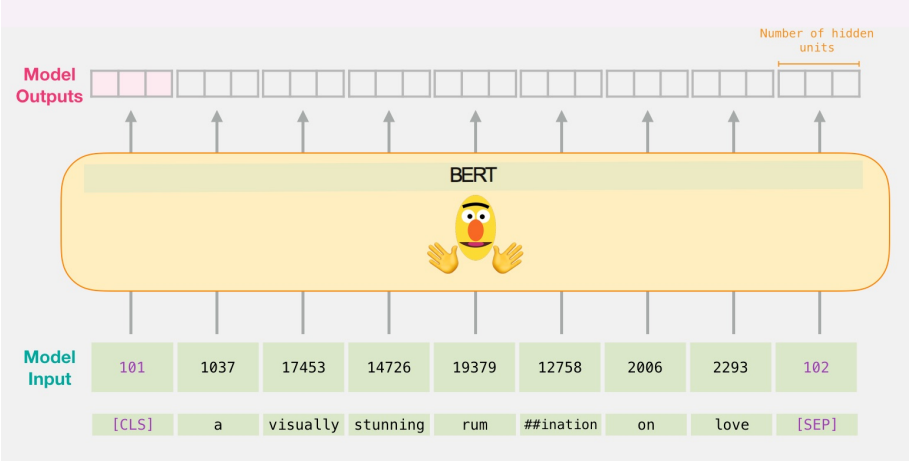
Slika 3.2: Pojednostavljen primjer klasifikacije slučajne šume

3.2 BERT

Prije nego nastavimo dalje, predstavimo biblioteku *Transformers*. Ona preuzima unaprijed pripremljene modele za zadatke već spomenutih procesa *NLU* i *NLG*. Biblioteka *Transformers* nudi tisuće prethodno treniranih modela za izvršavanje zadataka na tekstovima kao što su klasifikacija, traženje informacija, odgovaranje na pitanja, kreiranje sažetka, kreiranje prijevoda, itd. i to na više od 100 jezika. Cilj biblioteke je olakšati korištenje naprednog *NLP-a* svima. Ona što ona omogućuje je brzo preuzimanje i korištenje prethodno treninranih modela, koje korisnik lako prilagodi na svom skupu podataka.[17] Biblioteku *Transformers* podržavaju dvije popularne bibliote za dubinsko učenje, *PyTorch* i *TensorFlow*, s besprijekornom integracijom među njima, što omogućuje da se model trenira s jednom, a zatim zbog eventuane potrebe, učita u drugu.
BERT model (*Bidirectional Encoder Representations from Transformers*) razvijen je i predstavljen od strane Google-a a kreiran je 2018.godine. Njegovi kreatori su Jacob Devlin, Ming-Wei Chang, Kenton Lee i Kristina Toutanova. [16]
BERT bi najlakše mogli opisati kao tehniku strojnog učenja namijenjenu za NLP. Bazira se na modelu dubinskog učenja *Transformer* predstavljenog nešto ranije. Ponekad je zbog obima zadataka koje model izvršava kao i zbog memorije korištene za procesuiranje sam *BERT* "prevelik". Na sreću postoji niz "manjih" modela kao što su *DistilBert* (A *distilled* version of *BERT*), *RoBERTa* (*A Robustly Optimized*

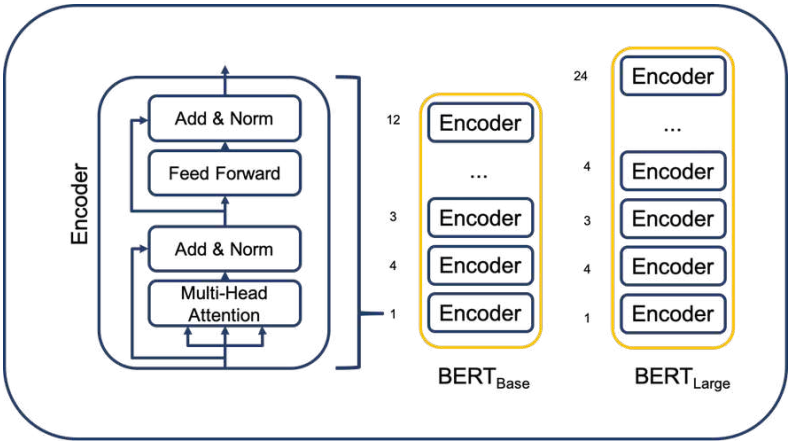
BERT Pretraining Approach), *ALBERT* (*A Lite BERT*) i razni drugi. Ti modeli su kreirani da olakšaju eventualne tehničke poteškoće pri korištenju *BERT-a* ali također da daju približne rezultate. Osim toga, sam *BERT* ima dvije varijante, *Base* i *Large*. Njihova glavna razlika je u tome što *Large* model koristi duplo više slojeva od *Base* modela pri kodiranju i dekodiranju teksta, kao i duplo dulji vektor u reprezentaciji određenog teksta. *BERT-ova* osnovna karakteristika je način na koji reprezentira tekst vektorima tj. kako tekst kodira u kod i kako kod dekodira opet u tekst, po potrebi (razne su primjene ovog modela).
Pokušajmo objansiti kako funkcionira. Model dobije kao ulaznu informaciju neku tekstualnu jedinicu. Ona može biti jedna riječ, niz riječi, jedna rečenica ili više rečenica. Zamislimo da mu kao ulaznu informaciju želimo proslijediti jednu rečenicu. Uvjet za to je tokenizacija te rečenice. O tokenizaciji generalno smo govorili već ranije, ali ovdje se pod tokenizacija misli korištenje već gotovog *tokenizatora* osmišljenog za ovaj model i prilagođenog verziji modela koji planiramo koristiti.
Sada vidimo da je ulazna informacija u model zapravo niz tokena, kojemu možemo dodati i tzv. *specijalne tokene*. U slučaju da u model šaljemo više rečenica, on će iza svake rečenice dodati specijalni token koji označava kraj rečenice *[102]* i prije svake rečenice specijalni token *[101]* koji označava početak te rečenice.Također u slučaju da nam je cilj klasifikacija, možemo i to naznačiti specijalnim tokenom *[CLS]* koji se formira na samom početku ulazne informacije, kao prvi token.Što se tiče duljine svake ulazne informacije, limit je 512 tokena (može se zadati da bude i manji). Iako je limitiran duljinom, razvijene su i neke tehnike koje podržavaju dulje tekstove kombinacijom dva ili više manjih modela, ali nama to neće biti potrebno.
Osim razdvajanja u tokene, zadatak *Tokenizatora* je da svaki token reprezentira s nizom brojeva. Te reprezentacije su preuzete iz unaprijed treniranih rječnika (spomenutih u odjeljku o reprezentaciji teksta) koji uzimaju u obzir sličnost riječi i njihovu povezanost.

Osnovni princip je da pripremljena reprezentacija teksta prolazi kroz višeslojno kodiranje. Broj slojeva se razlikuje od verzije do verzije. Osnovne dvije verzije *BERT-a*, *Base* i *Large* sastoje se, redom, od 12 i 24 sloja. Princip slojevitog kodiranja preuzet je iz *Transformer* modela. Svaki sloj (eng. *encode layer*) sastoji se od dva osnovna dijela, tzv. *Multi-head Attention* dijela i *Feed Forward Neuronskih mreža*. Kodiranjem, model u *Multi-head Attention* dijelu računa različite vektorske vrijednosti svakog tokena, i na kraju, njegov značaj u odnosu na druge tokene ulazne informacije. naziv *Multi-head* dolazi od toga što se cijeli postupak (*one head*) ponavlja više puta (*Base* 12, a *Large* 16 puta). Na kraju se rezultati različitih ponavljanja kodiranja dodaju jedan na drugog i množe sa matricom koja sadrži težine značajnosti određene kroz različita ponavljanja za svaki token tj. normalizira se da može ući kao ulazna informacija u idući sloj. Možemo reći da se u



Slika 3.3: Prikaz ulaznih i izlanih podataka.
Preuzeto sa [2]

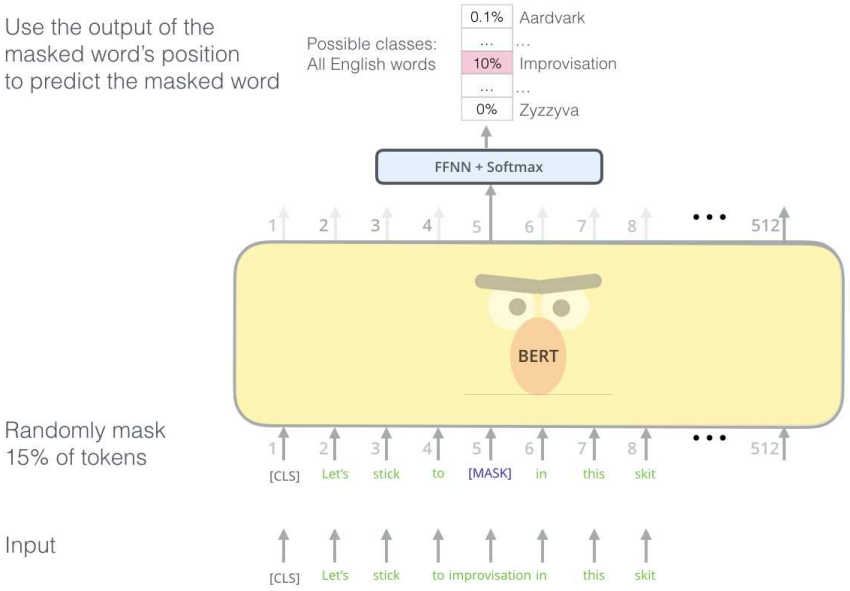
prvom sloju kodira token, u drugom se kodira kodirana vrijednost tokena itd. [3]



Slika 3.4: Arhitektura BERT-ovog algoritma za kodiranje.
Preuzeto sa [1]

Izlazna vrijednost posljednjeg sloja smatra se konačnom kodiranom vrijednosti ulazne informacije. Ono što je važno napomenuti je da osim što se *Base* i *Large* razlikuju po broju slojeva i broju ponavljanja postupka u svakom sloju, razlikuju se i u duljini izlaznog vektora koji predstavlja jedan token ulazne informacije. *Base* svaki token reprezentira vektorom duljine 768, dok *Large* svaki token reprezentira vektorom duljine 1024. Kada uzmemo u obzir da se samo u *Base BERT* modelu kodiranja teksta cijeli postupak kreiranja vektora duljine 768 ponavlja u 12 slojeva od kojih svaki po 12 puta ponavlja računanje nekoliko vektora vrijednosti ulazne informacije, i tako za svaki od 512 tokena, postaje jasno zašto su uvedene "manje obimne" verzije modela. Važno je napomenuti da sami dijelovi svakog sloja kodiranja nisu tako jednostavni i zahtijevali bi više stranica detaljne analize.
Ono što je zanimljivo je da ako planiramo *BERT* koristiti za klasificiranje, tada ne trebamo cijelu izlaznu vrijednost modela. Izlazna vrijednost dana je u obliku trodimenzionalnog tenzora, dimenzija $N \times 512 \times 768$ ili $N \times 512 \times 1024$ (ovisno koja je verzija u pitanju) gdje N predstavlja ukupan broj ulaznih informacija koje smo pustili u model, u našem slučaju broj izvještaja na kojima želimo istrenirati model. Prisjetimo se da možemo prvim tokenom u svakoj ulaznoj informaciji označiti da nam je svrha klasifikacija. Tada će nas na kraju interesirati samo izlazna vrijednost prvog tokena, tj, nećemo gledati ostalih 511 tokena, pa se dimenzija izlazne informacije u tom slučaju može svesti na $N \times 768$ ili $N \times 1024$. (Tako pripremljen dvodimenzionalni tenzor može se "poslati" u proizvoljni model klasifikacije, ili se mogu koristiti gotovi modeli biblioteke *Transformers* kao što su *BertForSequenceClassification*, *BertForMultipleChoice*, *DistilBertForSequenceClassification* i razni drugi).
Dosada smo objasnili kako funkcionira *BERT-ova* reprezentacija teksta. Preostaje pobliže opisati kako trenira podatke. Cijela poanta *Transformer* modela (time i *BERT-a*) je da se tekst ne čita "jednosmjerno", nego da se pokuša uhvatiti koncept cjeline čitajući od naprijed i natrag (*bidirectional*). Iako se u samom nazivu spominje "dvosmjerni", model ne ide konkretno jednom od naprijed jednom od natrag, nego sve tokene ulazne informacije dobije odjednom, za razliku od dosadašnjih modela koji bi uključivali tokene jedan po jedan u model.
Ako je ideja kodiranja potpuno preuzeta od *Transformer* modela, za koji se smatra da hvata kontekstualno značenje ulazne informacije, koja je razlika *Transformera* i *BERT-a*?
Ideja *BERT-a* koja ga razlikuje od svih dosadašnjih modela je u treniranju podataka. Model nasumično odabire 15% tokena od svake ulazne informacije i maskira ih. Da budemo precizni, od tih 15% tokena, 80% ih se zamijeni sa tokenom *[MASK]*, 10% ih se zamijeni sa nasumično odabranim tokenom, a preostalih 10% sa originalnim tokenom. Za svaki od tih 15% maskiranih tokena, algoritam pogađa koji bi na tom mjestu trebao biti originalni token. S tim da evaluacija tzv. funkcijom gubitka

(eng. *loss function*) uzima u obzir samo točnost predviđanja tokena zamijenjenih sa *[MASK]*, a ne i ostalih 20% maskiranih tokena. Pretpostavka je da se time konvergencija usporava, ali poimanje konteksta povećava.

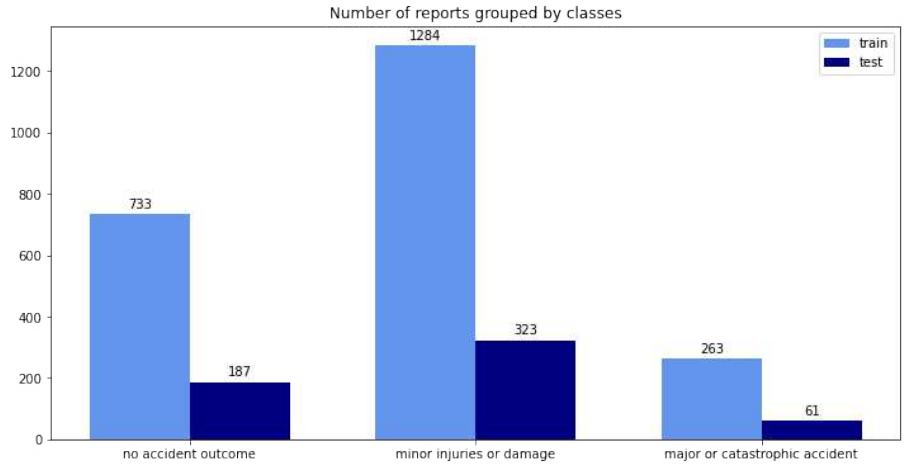


Slika 3.5: Prikaz maskiranja ulaznih informacija.
Preuzeto sa [2]

3.3 Evaluacija modela

Skup podataka je podijeljen na dva dijela, u prvi dio koji će biti korišten za treniranje modela spada 80% podataka, a u drugi, testni skup 20%. U oba skupa su raspoređeni izvještaji jednakog omjera po klasama. Testni skup podataka je onaj skup na kojemu ćemo isprobati model. Kada dobijemo predviđene klase za svaki od testnih izvještaja, usporediti ćemo predviđanje modela sa stvarnim podacima. Tako ćemo evaluirati model i dobiti u postotcima izraženu točnost i preciznost modela.

Usporedba predviđenih i stvarnih klasa prikazuje se tzv. konfuzijskom matricom. To je matrica u kojoj stupci označavaju stvarnu pripadnost klasi, a retci pripadnost modelom predviđenoj klasi. Model je bolji što se više predviđenih pripadnosti poklopilo sa stvarnim pripadnostima, tj. model je bolji što su vrijednosti na dijagonali



Slika 3.6: Primjer raspodjele izvještaja po klasama

konfuzijske matrice veće od ostalih vrijednosti. Uspješnost modela u postotcima, prema vrijednostima u konfuzijskoj matrici računamo na idući način:
-***Točnost*** (eng. *accuracy*) -udio točno klasificiranih izvještaja u skupu svih testnih izvještaja
-***Preciznost*** (eng. *precision*) -udio točno predviđenih izvještaja jedne klase u skupu svih (točno i netočno) predviđenih izvještaja za tu klasu
-***Odziv*** (eng. {recall) -udio točno predviđenih izvještaja jedne klase u skupu svih izvještaja koji zaista pripadaju toj klasi
-***F1-mjera*** (eng. *F1-score*) -za svaku klasu računamo vrijednost:

$$2 * (preciznost * odziv)/(preciznost + odziv)$$

Za potrebe daljnjeg računanja broj testnih izvještaja označimo sa n, pri čemu vrijedi $n = n_1 + n_2 + n_3$ gdje n_1, n_2 i n_3 označavaju stvaran broj izvještaja po klasama, prvoj, drugoj i trećoj, redom. Također uvedimo sljedeće oznake:

$$P_1 = preciznost\ 1.\ klase$$
$$O_1 = odziv\ 1.\ klase$$
$$F1_1 = F1\ mjera\ 1.\ klase$$

Analogno vrijede oznake za 2. i 3. klasu.
Sada pomoću izračunate uspješnosti modela zasebno po klasama, želimo izračunati ukupnu uspješnost modela. Dva su načina da to napravimo. Pomoću već izračunatih vrijednosti računamo makro i mikro (težinske) vrijednosti za evaluaciju:

$$\textit{Makro preciznost} = \frac{P_1+P_2+P_3}{3}$$
$$\textit{Makro odziv} = \frac{O_1+O_2+O_3}{3}$$
$$\textit{Makro F1} = \frac{F1_1+F1_2+F1_3}{3}$$
$$\textit{Mikro preciznost} = \frac{n_1*P_1+n_2*P_2+n_3*P_3}{n}$$
$$\textit{Mikro odziv} = \frac{n_1*O_1+n_2*O_2+n_3*O_3}{n}$$
$$\textit{Mikro F1} = \frac{n_1*F1_1+n_2*F1_2+n_3*F1_3}{n}$$

Osim navedenih, *BERT* model koristi i već spomenutu funkciju gubitka (eng. *loss function*). Što je bolji model, gubitak modela je manji. Ono što je korisno kao validator samog modela je omjer gubitka na treniranih podacima i gubitka na testnim podacima. Ako je omjer puno manji od 1, to ukazuje na tzv. *overfitting* ili pretreniranost, u značenju da je model previše prilagođen podacima na kojima je treniran i odskače od testnih podataka. U suprotnom, ako je omjer dosta veći od 1 onda uviđamo da nije dobro treniran.

Poglavlje 4

Eksperiment i rezultati

4.1 Random Forest

U ovom radu za klasificiranje ranije opisanih podataka, korišten je algoritam RandomForestClassifier biblioteke otvorenog tipa ***Scikit-learn***.[13]
Model za treniranje korišten je sa nizom unaprijed zadanih vrijednosti parametara. Neke od njih ćemo nabrojati:
n_estimators=100 -broj stabala koje model gradi
criterion="gini" -za mjeru efikasnosti korišten je Ginijev indeks
max_depth=None -stablo se grana dokle god čvorovi ne postanu homogeni odnosno dokle god u čvorovima ima dovoljno elemenata za podjelu
min_samples_split=2 -minimalan broj elemenata u čvoru potreban za novu podjelu
max_features="sqrt" -broj značajki koje se uzimaju u obzir prilikom odabira za podjelu čvora (uzima se $\sqrt{N}$ značajki gdje N označava ukupan broj značajki). U slučaju da se među odabranim značajkama ne nađe nijedna sa valjanom podjelom, algoritam će unatoč ovom parametru nastaviti potragu za dobrom podjelom.
Parametar koji nema zadanu vrijednost je:
class_weight="balanced" -težina klase se automatski prilagođava i to obrnuto proporcionalno frekvenciji klasa među ulaznim podacima
Eksperiment je napravljen na sljedeći način:
Podaci su se podijelili (ravnomjerno po klasama) u omjeru 80:20. Gore navedeni algoritam slučajne šume koristio se kao model i treniran je na prvom skupu (80% podataka predodređenih za treniranje). Na preostalih 20% se model testirao i računale su se evaluacijske vrijednosti *preciznost*, *odziv* i *F1-mjera*. Cijeli postupak podjele podataka, treniranje modela i evaluacija modela na skupu za testiranje, ponovljen je 10 puta. *Preciznost*, *odziv* i *F1-mjera* modela određena je kao aritmetička sredina 10 uzastopnih rezultata.

Vrijednosti u pojedinim etapama eksperimenta prikazani su u tablici:

i	**Preciznost**	**Odziv**	**F1-mjera**
0	0.448766	0.425037	0.436579
1	0.468026	0.433118	0.449896
2	0.459941	0.41805	0.437996
3	0.485935	0.433894	0.458442
4	0.484795	0.433319	0.457614
5	0.461311	0.418567	0.438901
6	0.43408	0.410745	0.42209
7	0.439063	0.40324	0.42039
8	0.456106	0.422449	0.438633
9	0.476228	0.424319	0.448777

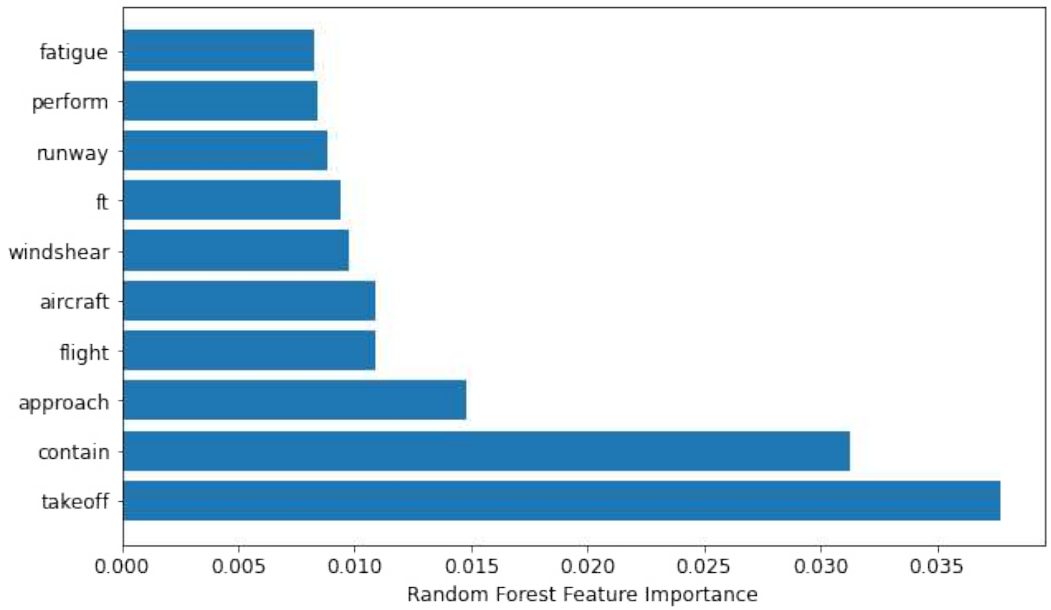
Prosječni rezultati modela po klasama:

	Preciznost	**Odziv**	**F1-mjera**
no accident outcome	0.48192	0.43478	0.45713
minor injuries or damage	0.62435	0.74844	0.68079
major or catastrophic accident	0.28571	0.0895	0.1363

Ukupni rezultati modela:

Točnost	0.57	
Preciznost	**Odziv**	**F1-mjera**
0.461766	0.4222738	0.4409318

S obzirom na lošije rezultate može nas zanimati koje su se značajke pokazale važnije u modelu. *RandomForestClassifier* ima atribut *feature_importances_* (eng. za važnost značajki) koji je za 10 najvažnijih značajki dao ovakav graf na slici 4.1. Primjetimo da je graf dosta sličan onom grafu sa slike 2.3 koji prikazuje učestalost riječi po izvješćima.



Slika 4.1: Najvažnije značajke Random Forest modela

4.2 BERT

Za BERT reprezentaciju i klasifikaciju korišteni su modeli:
$BertTokenizer[frompretrained](\text{"}bert-base-uncased\text{"})$
$BertForSequenceClassification$
$DistilBertTokenizer[frompretrained](\text{"}distilbert-base-uncased\text{"})$
$DistilBertForSequenceClassification$

Zbog potrebne memorije korišten je DistilBert umjesto BERT modela kada je model testiran na duljim tekstovima. *From pretrained* u definiciji tokenizatora znači da je rječnik po kojem se radi reprezentacija treniran unaprijed na velikim bazama tekstova, kao što je spomenuto ranije. Oba modela preuzeta su od biblioteke Transformers koja je otvorenog tipa. *DistilBertForSequenceClassification* predstavlja već gotovi model koji po BERT reprezentaciji klasificira željene tekstove.
Ono što je važno napomenuti je da BERT ima zadanu formu ulaznih podataka. Osim što prima kompletne tekstove, prima i neke dodatne informacije o njima. Za početak napomenimo kako sami možemo odabrati koju duljinu teksta želimo promatrati, maksimum je 512 tokena. Jedan token predstavlja jednu tekstualnu jedinicu, uglavnom je to jedna riječ, no ako je riječ manje poznata (manje korištena u unaprijed istreniranom rječniku) tada će se riječ rastaviti na poznatije dijelove.

Npr. na slici 3.3 možemo primijetiti da je riječ *rumination* tokenizator rastavio na dva tokena *rum* i *ination* sa obilježjem # ispred tokena koji je nastavak prethodnog tokena.
Ako proučavamo tekstove koji imaju manji broj riječi (ili tokena) od zadanog maksimuma, možemo zadati i manju duljinu i to proizvoljno (npr. 100, 128, 256...). Pokazalo se da model nekada daje uspješne rezultate čak i ako dio tekstova nije prikazan cijeli. Međutim što se događa sa onim tekstovima koji su manji ili veći od zadane duljine? Ako je broj tokena u jednom teksu manji od zadane duljine, na ostala mjesta se nadodaje 0, a ako je dulji onda ga se naprosto odsječe na zadanoj duljini. Listu ulaznih podataka koji sadrži vektore tokena određene duljine (po potrebi dopunjene s nulama) zovemo *input ids*. Svakako treba računalu na neki način označiti koji dio vektora reprezentacije označava tekst a koji je nadopunjen nulama. Zato je uvedena varijabla *input mask* koja sadrži listu vektora. Svaki vektor je jednake duljine kao i vektori u *input ids* a pamti informacije o tokenima, koji je originalno iz teksta, a koji je nadodan zbog duljine. Ako je neki vektor *input ids* bio duljine 512, od čega je samo prvih 300 tokena bilo iz originalnog teksta a preostalih 212 su nadodane 0, tada će pripadajući vektor *input mask* na prvih 300 mjesta imati 1 a na ostalih 212 0.
Osim navedenih ulaznih podataka spomenuti ćemo i *segment ids* koji DistilBert, za razliku od BERTA nema. Riječ je oznakama tokena iste vrste rečenice sa 0 ili 1, ovisno kojoj vrsti rečenice pripada, npr. kada se model koristi u svrhu pronalaženja odgovora na pitanje, onda će se model trenirati na nizu parova pitanje/odgovor rečenica od kojih će se odgovarajući vektor *segment ids* za tokene pitanja označiti sa 0, a za tokene odgovora sa 1 ili obrnuto. S obzirom da mi koristimo model za klasifikaciju i podatke ne grupiramo u parove rečenica, pri korištenju BERT modela, svi *segment ids* vektori sadržavali bi samo broj 1 tako da nam njihov izostanak ne utječe na model. Ulazne podatke potrebne za BERT možemo napraviti sami ili to prepustiti funkciji tokenizatora kojoj kao parametre pošaljemo:

$$\begin{gathered} maxlength = 128 \text{ -zadavanje maksimalne duljine teksta} \\ truncate = True \text{ -skraćivanje duljih tekstova na zadanu duljinu} \\ padding = True \text{ -proširivanje kraćih tekstova do zadane duljine} \end{gathered}$$

Što se tiče evaluacije, sam model preuzet iz biblioteke Transformers ima argument *eval* za evaluaciju kojim računa uspješnost na testnim, dosad neviđenim podatcima. Prije samih rezultata važno je još samo objasniti nekoliko parametara koji utječu na treniranje modela. Kada zadajemo model moramo zadati dodatno koliki je *batch size*, *number of epochs* i *learning rate*.
Broj epoha (eng. *number of epochs*) označava koliko puta ćemo napraviti cijeli

proces, npr. ako je broj epoha = 10 tada će model 10 puta proći proces treniranja i svaki put računati evaluaciju na validacijskim podatcima (dodatni dio podataka kojeg odvojimo od podataka za treniranje modela da nam služi kao testiranje određenih parametara koje želimo poboljšati kroz model). Kroz epohe se očekuje da će rezultati biti sve bolji, međutim ako nakon određenog broja epoha ne dolazi do značajne promjene u rezultatima tada nema smisla računati veliki broj epoha, dovoljno je ići do razine gdje se postižu spomenuti rezultati.
Veličina uzorka (eng. *batch size*) određuje koliko će se podataka obrađivati odjednom. Npr. ako je $batchsize = N$ (gdje je N broj tekstova koje želimo obraditi modelom) tada će u svakoj epohi model odjednom obraditi sve dane ulazne podatke, međutim ako je $batchsize$ = 16, 32 ili 64 (često korištene veličine za ovaj parametar) tada će se u svakoj epohi podatci obrađivati po skupinama od $\lfloor \frac{N}{batchsize} \rfloor$tekstova, osim zadnje skupine koja će sadržavati ostatak tekstova. Osim što je to jedan od načina uštede memorije, između svake skupine će se prilagođavati težine koje predstavljaju razinu značajnosti riječi i odnosa među njima. Što nas dovodi do pojma *learning rate* ili tempo učenja. To je oznaka za brzinu kojom se težine u modelu prilagođavaju. Dopuštena količina promjene parametara između dva računanja. Poznata je činjenica da se premalom brzinom učenja rezultati postižu sporije, a prevelikom može uzrokovati velike oscilacije među rezultatima između svake prilagodbe težina. Najčešće se zadaje kao mali broj, od e^{-1} do e^{-6}.

Rezultati DistilBert modela sa dolje navedenim parametrima:

$$maxlength = 128$$
$$numberofepochs = 5$$
$$batchsize = 16$$
$$learningrate = e^{-5}$$

Točnost	0.66		
	Preciznost	**Odziv**	**F1-mjera**
no accident outcome	0.55	0.55	0.55
minor injuries or damage	0.72	0.79	0.75
major or catastrophic accident	0.29	0.08	0.13
Težinski prosjek	0.63	0.66	0.64

Dobiveni rezultati su lošiji od očekivanog i od prijašnjih rezultata na nekim drugim podatcima dobivenih ovim modelima. Za pretpostaviti je da zbog različite kvalitete početnih tekstualnih podataka, koja automatski utječe na kvalitetu njihove reprezentacije (tj. ulaznih podataka modela) i sami rezultati modela poprilično osciliraju.

Kodovi kojima su rađeni eksperimenti i dobiveni rezultati nalaze se na poveznici:

`https://github.com/LukrecijaTudor/NLP`

Osim kodova korištenih za ovaj rad, na gore navedenoj poveznici nalaze se i eksperimenti vezani za klasterizaciju i modeliranje različitih tema kroz izvještaje u skupu podataka (eng. *clustering* i *Topic Modeling*). Te metode su dio strojnog učenja i spadaju u metode nenadziranog učenja (ne uzima se u obzir oznaka rizika koja je dodijeljena svakom izvještaju niti ikakva druga dodijeljena oznaka već se s obzirom na izračunate sličnosti izvještaji grupiraju u tzv. *klastere*, te im se određuju najučestalije riječi ili obilježja koja ih karakteriziraju).

Poglavlje 5

Zaključak

Novije metode računalnog procesuiranja i razumijevanja teksta svakako pokazuju znatan napredak u odnosu na prijašnje metode i alate. No sami algoritmi, koliko god kompleksno i detaljno razvijeni, nisu dovoljni da njihova primjena rezultira željenim uspjehom, makar postoje ogledni primjeri sa zavidnim rezultatima. Kao što je to i u životu čest slučaj, informacije koje dobijemo i podatci koje koristimo nisu idealni. Nisu idealno izbalansirani, ili nisu dovoljno detaljni, nekad ih je premalo, nekad sadrže raznorazne greške (pisane, izgovorene, izbrisane, zaboravljene...) i sl. Iz tog razloga smo naučeni istraživati razne načine da podatke prilagodimo računalu bez da im mijenjamo smisao. Također, znamo da računalo razumije brojeve. I prema tome se pokazalo uspješnim, ako nemamo prave brojeve, nekim algoritmom aproksimirati dovoljno bliske brojeve da bi metoda bila što uspješnija. U tome dodatno prednjače metode strojnog učenja jer su konstruirane tako da računalo svakom iteracijom samo računa sve bolje i bolje aproksimacije željenih parametara. Nažalost, tekst ne možemo tek tako predočiti brojevima. Slova, slogovi, riječi i njihove kombinacije, makar bili jednaki lingvistički, kontekstualno daju drugačije značenje. Kod brojeva to nije tako. Ako znamo u kojem smo brojčanom sustavu, znamo i njegove zakonitosti. Na primjer, broj 1 ima potpuno drugačije značenje u binarnom i dekadskom brojevnom sustavu. Ali ako znamo u kojem smo onda znamo i kako protumačiti 1. Međutim za tekstualne podatke nije dovoljno samo uzeti jezik kojim su podaci izrečeni/napisani. Sam jezik ima bogatu strukturu i sadrži razne unutarnje sustave. Uzmimo za primjer riječi (Jadransko) *môre*, (noćne) *mòre* i *mȍre* (u značenju *može* na dalmatinskom dijalektu). Ono po čemu se ove tri riječi razlikuju je naglasak. Mogli bismo tekst, osim po jeziku razlikovati i po narječjima, ili padežima, ili možda čuvati informacije i o naglasku riječi. Međutim moguće je i da, ako metodama razumijevanja teksta procesuiramo nečiji govor, neki naglasak može biti krivo izrečen a dijalekt se može razlikovati

čak u mjestima udaljenim samo par kilometara i sl. Cilj ovog primjera je dati uvid u to koliko faktora ima utjecaj na kvalitetu podataka. U tom smjeru, čini mi se da bi generalni napredak računalnih metoda procesuiranja i razumijevanja teksta mogao biti baš u tome kako reprezentiramo podatke računalu. Kada bi se modeli namijenjeni klasifikaciji avijacijskih izvještaja trenirali na zasebnim bazama tekstova, pisanih isključivo za potrebe aviacije, vjerojatno bi rezultati bili puno bolji. No, to je tehnički malo zahtjevnije. Ako model treniramo na podatcima sa Wikipedije tada smo ga naučili jednom općem znanju, o svemu pomalo. Logičan idući korak je napraviti usko specijalizirane rječnike ovisno o potrebama upotrebe modela.
Ipak, možda ni to nije idealno rješenje i možda trebamo sasvim promijeniti perspektivu. Što ako nije dovoljno predočiti riječi brojevima i pokušati ih uklopiti u poznate i postojeće brojčane sustave. Možda je potrebno kreirati novi sustav specijalno za tekstualnu analizu, s nekim novim, dosad zanemarenim obilježjima (npr. razlika zbog naglasaka kao u primjeru).
Podataka je sve više, informacijama smo obasuti sa svih strana, metode strojnog učenja čine bitan faktor tehničkog napretka i razvoja svijeta, a procesuiranje teksta je perspektivna grana čitavog AI sustava. Dio tehnološke budućnosti leži upravo na ovim naprednijim modelima i budućim rješenjima što bolje njihove implementacije. Nadam se da ćemo i sami uskoro svjedočiti novim čudima tehnologije, a još više se nadam da se te metode neće iskoristiti u krive svrhe.
Što se tiče ciljeva ovog rada, veliki potencijal napretka zračne sigurnosti leži u metodama procesuiranja i razumijevanja teksta, jer su opisi događaja, bilo u pisanom ili još bolje u glasovnom zapisu najbolji uvid u ono što se tokom leta događa, koje su moguće nepravilnosti, njihovi uzroci, posljedice ili eventualno načini sanacije koji kasnije drugima služe kao primjer. Kaže se da se najbolje uči iz iskustva, a ovim pristupom se omogućuje računalna obrada i razmjena velikog broja ljudskih iskustava, što detaljnija i što pažljivije interpretirana iskustva to bolja ulazna informacija za računalo.

Bibliografija

[1] pristupljeno 2020. URL: https://humboldt-wi.github.io/blog/tags/bert/.

[2] Jay Alammar. *Finding the Words to Say: Hidden State Visualizations for Language Models*. pristupljeno 2020. URL: http://jalammar.github.io.

[3] Jay Alammar. *The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)*. pristupljeno 2020. URL: http://jalammar.github.io/illustrated-bert/.

[4] Arghya Bhattacharya. *NLP and text mining: A natural fit for business growth*. pristupljeno 2020. URL: https://www.sentisum.com/library/nlp-and-text-mining.

[5] Leo Breiman. "Random Forests". In: *Statistics Department University of California Berkeley, CA 94720* (Siječanj 2001.).

[6] B.J. Copeland. *Artificial-intelligence*. pristupljeno 2020. URL: https://www.britannica.com/technology/artificial-intelligence.

[7] EASA. 2014. URL: https://www.easa.europa.eu/home.

[8] *Easy Access Rules for Occurrence Reporting (Regulation (EU) No 376/2014)*. 2014. URL: https://www.easa.europa.eu/sites/default/files/dfu/easy_access_rules_for_occurrence_reporting_regulation_eu_no_376-2014.pdf.

[9] Eda Kavlakoglu. *NLP vs. NLU vs. NLG: the differences between three natural language processing concepts*. 2020. URL: https://www.ibm.com/blogs/watson/2020/11/nlp-vs-nlu-vs-nlg-the-differences-between-three-natural-language-processing-concepts/.

[10] Will Koehrsen. *An Implementation and Explanation of the Random Forest in Python*. 2018. URL: https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76.

[11] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. 2012.

[12] Artem Oppermann. *What is Deep Learning and How does it work?* 2019. URL: https://towardsdatascience.com/what-is-deep-learning-and-how-does-it-work-2ce44bb692ac.

[13] *Scikit-Learn biblioteka*. pristupljeno 2020. URL: https://scikit-learn.org/stable/index.html.

[14] Luis G. Serrano. *Grokking Machine Learning, chapter 7.* Manning Publications, 2019., cjelovito izdanje se očekuje 2021.

[15] *spaCy biblioteka*. pristupljeno 2020. URL: https://spacy.io.

[16] Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2018. URL: https://arxiv.org/abs/1810.04805.

[17] *Transformers biblioteka*. pristupljeno 2020. URL: https://huggingface.co/transformers/index.html.

[18] Jaime Zornoza. *Random Forest Explained*. 2020. URL: https://towardsdatascience.com/random-forest-explained-7eae084f3ebe.

Sažetak

Let avionom civilizacijski je postao svakodnevni dio života. Naravno to ne znači da su tehnološki i organizacijski sustavi koji zračni promet čine mogućim i održivim, trivijalni. Ono na što se najviše obraća pozornost su poteškoće u tehničkom pregledu, pri polijetanju, slijetanju i tokom samog leta. Zbog toga je od strane EAS-e (eng. European Aviation Safety Agency) uvedena odredba o obaveznom izvještavanju svakog nepredviđenog događaja. Izvještaje uglavnom pišu piloti ili idući odgovorni u hijerarhiji leta i to najkasnije 72 sata nakon leta. U svrhu što brže organizacije i klasifikacije tih dokumenata s obzirom na razinu rizika u ovom radu su prikazani eksperimenti primjene različitih klasifikacijskih algoritama i popratnih nužnih metoda reprezentacije na skupu od nešto manje od 3000 zvještaja aviokompanije Adria Airways na engleskom jeziku. Izvještaji su, po unaprijed određenim parametrima, klasificirani prema faktoru rizika koji je dan u neekvidistantnoj skali od 1 do 2500. S obzirom na taj faktor, postoje 3 klase događaja:
-Događaj bez nesreće u ishodu
-Događaj s minimalnom nesrećom ili štetom
-Događaj s katastrofalnom nesrećom (uključuje smrtni slučaj)
Kako računalu ne možemo dati čisti tekstualni podatak na analizu, korištene su metode tekstuane reprezentacije onim što računalo razumije, a to su brojevi.
Reprezentacije teksta korištene u ovom radu su TF-IDF (Term Frequency-Inverse Document Frequency) bazirana na klasičnoj BoW (Bag of Words) reprezentaciji i BERT kodiranje teksta (Bidirectional Encoder Representations from Transformers) koje se temelji, kao što ime kaže, na starijoj reprezentaciji modela Transformers, ali je strukturno šire i konceptualno razvijenije.
Na gotovim tekstualnim reprezentacijama korišteni su klasifikacijski modeli. Opisani, i u eksperimentu isporbani modeli su RandomForestClassifier biblioteke otvorenog tipa Scikit-learn i model DistilBertForSequenceClassification, biblioteke Transformers koja je također otvorenog tipa.
Rezulati nijednog modela nisu pretjerano uspješni, najbolje postignut rezultat klasifikacije određen mjerom F1 je 64% ukupno i 75% samo za klasu događaja s minimalnom nesrećom ili štetom. Iako rezultati nisu izvrsni, vidljiv je znatan napredak

od modela RF do modela BERT. Jedan od razloga loših rezultata je zasigurno nebalansiran skup podataka. Najviše je izvještaja klase rizika od minimalne nesreće, pa klasa događaja bez nesreće u ishodu,a najmanje onih sa katastrofalnim ishodom. S obzirom da je pisanje izvještaja obavezno samo u slučaju nepredviđenog događaja, a rizika od katastrofalnih ishoda, na sreću, nema toliko puno, nebalansiranost podataka je logična. Drugi mogući uzrok loših rezultata je to što osim jako dobrih modela, unatoč nastojanjima i velikom uspjehu na tom polju, reprezentacije teksta i dalje ne mogu pouzdano uhvatiti koncept. Osim razlika u značenju isto napisanih riječi, razlike mogu biti u dijalektu, u naglascima i slično. Na primjer riječi (Jadransko) *môre*, (noćne) *mòre* i *mȍre* (u značenju *može* na dalmatinskom dijalektu) imaju tri potpuno različita značenja a ovisno koliko je dobar rječnik reprezentacije, računalo će te tri riječi pročitati možda isto, možda samo slično, možda čak i različito, ali to ne mora biti pravilo. Možda bi cijeli sustav čitanja, slušanja, razumijevanja i procesuiranja teksta trebao biti konstruiran neovisno o brojevnim reprezentacijama rječnika.
Svakako, veliki potencijal napretka zračne sigurnosti leži u metodama procesuiranja i razumijevanja teksta, jer su opisi događaja, bilo u pisanom ili još bolje u glasovnom zapisu najbolji uvid u ono što se tokom leta događa.

Summary

Travelling by plane has become part of our everyday life. Of course, this does not mean that the technological and organizational systems that make air transport possible and sustainable are trivial. What is most noticeable are the difficulties in the technical inspection, during take-off, landing, and during the flight itself. Therefore, EASA (European Aviation Safety Agency) has introduced a provision for mandatory reporting of any unforeseen events. Reports are generally written by pilots or the next in line of responsibility in the flight hierarchy no later than 72 hours after the flight. In order to organize and classify these documents as quickly as possible, according to the level of risk, this paper presents experiments with the application of various classification algorithms and accompanying necessary methods of representation on a set of fewer than 3000 reports of Adria Airways in English. These reports, according to predetermined parameters, are classified in line with the risk factor given in the non-equidistant scale from 1 to 2500. Considering this factor, there are 3 classes of events: - No accident outcome -Minor injuries or damage - Major or catastrophic accident (including death) Since we cannot give a computer pure textual data for analysis, methods of textual representation by what the computer understands are used, and these are numbers. The text representations used in this paper are TF-IDF (Term Frequency-Inverse Document Frequency) based on the classical BoW (Bag of Words) representation and BERT text encoding (Bidirectional Encoder Representations from Transformers) which is based, as the name suggests, on the older representation of the Transformers model, but is structurally broader and conceptually more developed. Classification models were used on the finished textual representations. The models described and used in the experiment are the RandomForestClassifier open source library Scikit-learn and the model DistilBertForSequenceClassification, a Transformers library that is also open source. The results of none of the models are overly successful, the best-achieved classification result determined by measure F1 is 64% for all classes and 75% for class with minor injuries or damage. However, significant progress is visible from the RF model to the BERT model. One of the reasons for invalid data is certainly an unbalanced data set. Most reports are of the class with a minimal accident, followed

by events without an accident in the outcome, and the least of those with the ruinous outcome. Given that, writing a report is mandatory only in case of an unforeseen event, and fortunately, there are not so many events with risk of disastrous outcome, the imbalance of data is logical. Another possible cause of poor results is that, apart from very good models, despite efforts and great success in this field, text representations still cannot reliably grasp the concept. In addition to differences in the meaning of the same words, differences can be in dialect, accents, and the like. For example, the words (reading) a *book* and to *book* a flight have completely different meanings, and depending on how well the vocabulary is trained, the computer will maybe read those words the same, maybe just similar, maybe even different, but that doesn't have to be the rule. Perhaps the whole system of reading, listening, understanding, and processing a text should be constructed independently of the numerical representations of the dictionary. Certainly, the great potential of air safety advances lies in text processing and comprehension methods, as descriptions of events, whether in written or even better in voice recording, provide the best insight into what happens during the flight.

Životopis

Lukrecija Tudor (07.02.1991.) rođena je i odrasla u Splitu gdje 2009. godine završava Prirodoslovno-matematičku gimnaziju paralelno pohađajući Dramski studio Gradskog kazališta mladih. Iste godine, po završetku srednje škole upisuje studij glume na Akademiji dramskih umjetnosti u Zagrebu. 2012. godine postaje prvostupnicom glume, a 2014. završava i diplomski studij te stiče titulu mag.art. Iako tokom i nakon studija sudjeluje u raznim kazališnim, filmskim i tv produkcijama, 2015.godine upisuje studij Matematike na Prirodoslovno-matematičkom fakultetu u Zagrebu, a 2018. na istom fakultetu upisuje i diplomski studij Matematičke statistike. Trenutno radi kao projektant u BI odjelu Raiffeisen banke.