SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
MATEMATIČKI ODSJEK

Ana Paliska i Filip Srnec

Prepoznavanje glazbenih akorda koristeći tehnike strojnog učenja

Zagreb, 2017.

Ovaj rad izrađen je na na Matematičkom odsjeku Prirodoslovno-matematičkog fakulteta, pod vodstvom dr. sc. Tomislava Šmuca i doc. dr. sc. Goranke Nogo, i predan je na natječaj za dodjelu Rektorove nagrade u akademskoj godini 2016./2017.

Sadržaj

Poglavlje 1

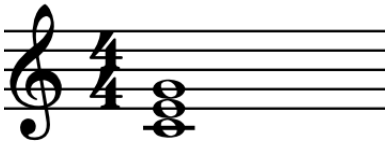
Uvod

Glazba je svuda oko nas. Od najranijih ljudskih početaka, ona se koristila i primjenjivala u razne svrhe, a najviše od svega je vezana uz veselje i uživanje. Osnovni dio nekog glazbenog djela je melodija i to je ono po čemu je neko djelo prepoznatljivo i posebno. No, glazbeno djelo je u većini slučajeva mnogo složenije. Više različitih tonova je spojeno u cjeline stvarajući harmonije koje glazbeno djelo upotpunjuju. Akord je pojam koji označava tri ili više tonova koji su odsvirani ili otpjevani istodobno. U ovom poglavlju bit će predstavljen i opisan problem prepoznavanja glazbenih akorda te njegova primjena. Detaljnije će biti pojašnjeni pojmovi iz teorije glazbe koji su važni za daljnje praćenje rada.

1.1 Problem prepoznavanja akorda

Osnovni element koji se promatra u grani teorije glazbe koja se naziva harmonija (grč. *harmonia* što znači sklad, slaganje) je *akord*. Sam pojam akord označava tri ili više tonova koji su otpjevani ili odsvirani ("zvuče") istodobno ili su odsvirani ili otpjevani odvojeno, ali pritom ostavljaju dojam suzvučnosti.

Akordi se u teoriji glazbe formalno mogu klasificirati na nekoliko različitih načina: početni ton, intervali (razmaci) između tonova koji se pojavljuju u samom akordu, stupnjevi u pripadnoj ljestvici koji se pojavljuju u akordu i drugi (za detalje vidi [26]). Nadalje, svakom akordu je moguće pridružiti odgovarajuću dogovorenu oznaku koja glazbenicima daje do znanja koji akord u nekom trenutku treba biti odsviran. Takva notacija nije strogo formalna pa se samim time ne pojavljuje u strogo formaliziranim notnim zapisima jer često oznaka akorda zapravo obuhvaća cijelu klasu akorda koji odgovaraju toj oznaci (vidi Primjer 1). U današnje vrijeme, ta notacija se često



Slika 1.1: Kvintakord dura na tonu C, oznaka akorda CM[2]

koristi jer osigurava glazbenicima zajednički jezik koji olakšava međusobnu komunikaciju i daje osnovnu informaciju o harmonijama koje se pojavljuju u promatranoj pjesmi.

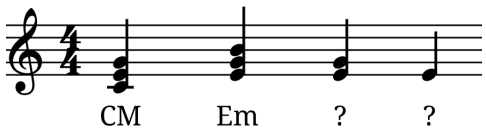
Primjer 1. *Kao primjer promotrimo akord pod nazivom kvintakord dura na tonu C*[1] *koji se sastoji od tonova C, E i G (vidi Slika 1.1). Tom akordu se pridružuje oznaka CM ili samo C koja označava da je pojava tonova C, E i G karakteristična za C dur.*

Jedan od osnovnih problema vezanih uz područje procesiranja i analize glazbe je problem automatskog prepoznavanja akorda. Ideja je da za neko glazbeno dijelo zapisanom u određenom formatu kao ulaz, na izlazu dobijemo odgovarajuće oznake akorda koje daju informaciju u kojem se trenutku mijenjaju akordi i koje su to promjene.

Iako sama teorija glazbe jasno određuje pravila kako je neki akord građen (vidi [26] ili [30]), za određivanje tipa (vrste) odgovarajućeg akorda u određenom vremenskom trenutku takav pristup ne mora dati jedinstvenu procjenu. Određivanje glazbenog akorda je puno širi problem. Naime, potrebno je promatrati i *kontekst* u kojem se promatrani skup tonova pojavljuje jer tonovi u različitom kontektsu imaju različite uloge. U nekim slučajevima, kada samom teorijskom analizom dolazi do više mogućnosti, kontekst postaje od presudne važnosti (vidi Primjer 2). Kontekst je pojam koji je teško egzaktno definirati. Pod pojmom konteksta se može smatrati tonalitet u kojem je glazbeno dijelo skladano, akordi koje se pojavljuju neposredno prije i neposredno nakon promatranog akorda, naglašenost pojedinih tonova u akordu, načini prijelaza između akorda, vrijeme nastanka glazbenog djela i mnoge druge karakteristike. Jedan od radova na sličnu temu gdje se primarno promatrao kontekst u kojem se akordi pojavljuju je [24]. Rješavanje problema prepoznavanja akorda uvelike ovisi o mogućnosti opisivanja danog konteksta u kojem se neki akord pojavljuje i uključivanja tog konteksta u sam proces prepoznavanja. U nastavku rada ćemo koristiti neke pojmove i tvrdnje iz

[1] *Kvintakord dura* je trozvuk građen je od intervala *velike terce* i *čiste kvinte* u odnosu na početnu notu.

[2] Notni zapisi prikazani u ovom radu generirani su koristeći *online* alat dostupan na `https://flat.io`



Slika 1.2: Višeznačnost određivanja oznake akorda bez poznavanja konteksta

domene problema pod pretpostavkom da su čitatelju jasni iz konteksta u kojem se spominju. Neke tvrdnje iz teorije glazbe možda neće biti dovoljno detaljno pojašnjene, no njihov detaljniji opis bi izlazio iz okvira ovog rada. Pozivamo čitatelja da odgovore na moguća pitanja pronađe u [26], [30], [6] ili u nekoj drugoj literaturi koja se bavi osnovnim pojmovima teorije glazbe i harmonije.

Primjer 2. *Pretpostavimo da kao vremenski interval kojem pridružujemo oznaku akorda uzimamo jednu četvrtinku u četveročetvrtinskoj mjeri (jednu dobu u taktu od četiri dobe; jedan promatrani događaj je četvrtina jednog takta). Zamislimo da se nekom promatranom intervalu u promatranoj skladbi pojavljuju tonovi E i G. U Primjeru 1 naglašeno je da je kvintakord dura na tonu C građen od tonova C, E i G. Uočimo da se u tom akordu pojavljuju tonovi E i G, kao i u našem promatranom primjeru. No, isti se tonovi, između ostalog, pojavljuju i u kvintakordu mola na tonu E u kojem se pojavljuju tonovi E, G i B*[3]*. Uočimo da je bez dodatne informacije nemoguće odrediti koju oznaku akorda treba pridijeliti tom vremenskom trenutku: CM ili Em (označava E mol). Što ako je odsviran samo ton E? U tom slučaju situacija postaje još kompliciranija. Opisana situacija u notnom zapisu prikazana je na Slici 1.2. Napomenimo da u ovom radu koristimo oznake akorda koje se koriste u popularnoj glazbi ili njihove manje modifikacije*[4]*, a ne službene oznake koje se koriste u teoriji glazbe. Za detalje o oznakama akorda koji se koriste u glazbi vidi [6].*

S obzirom na sve navedeno, problem prepoznavanja akorda je vrlo specifičan u smislu izgradnje skupa podataka koji se može koristiti prilikom učenja modela. Naime, označavanje stvarnih primjera je izrazito zahtjevan posao koji iziskuje veliku količinu ljudskog znanja i iskustva. Glazba se uvijek može shvaćati na različite načine, ona je gotovo uvijek barem u nekoj mjeri

[4]Na primjer, akord *C dura* se češće zapisuje samo kao *C*, a ne kao *CM* kao u ovom radu, no radi konzistentnosti sa skupom podataka s kojim radimo (vidi Poglavlje 2) te jednoznačnosti prilikom referenciranja na ton, a ne akord, *C* odabrali smo ovakvu notaciju.

[4]U hrvatskoj literaturi ton *B* se obično označava s *H*, no zbog notacije u originalnim podacima te standardom koji se koristi literaturi na engleskom jeziku u ovom radu koristit ćemo oznaku *B*.

subjektivna. U prethodnim odlomcima već je naznačeno da je gotovo nemoguće pronaći 1-1 korespodenciju između formalnih pravila u teoriji glazbe te oznaka akorda. Različite osobe čuju različite stvari u istoj glazbi pa oznaka može isključivo biti odraz "istine" kako ih vidi osoba koja na tome radi. Samim time, ne postoji mnogo provjerenih javno dostupnih skupova podataka koji se mogu koristiti za usporedbu rezultata.

1.2 Ciljevi rada

Problem automatskog prepoznavanja akorda može se interpretirati kao klasifikacijski problem strojnog učenja. Cilj ovog rada je ispitati nekoliko pristupa za prepoznavanje akorda pomoću tehnika strojnog učenja. Rad smo podijelili u nekoliko faza koje se međusobno nadograđuju. Tijekom tih faza povećavali smo kompleksnost modela koji opisuje promatrani problem. Dobiveni rezultati su statistički analizirani i interpretirani u domeni promatranog problema.

Ono na što smo se fokusirali je prepoznavanje svih akorda neke pjesme u konkretnom vremenskom trenutku, onako kako se oni sekvencijalno pojavljuju u samoj pjesmi. Motivacija za takav pristup je ideja izgradnje preciznog sustava za prepoznavanje glazbenih akorda koji će biti u stanju samostalno klasificirati akorde u proizvoljnoj skladbi. U zadnjih nekoliko godina razvija se sve više sustava tog tipa, no oni još uvijek nemaju adekvatnu preciznost. Vjerujemo da je pristup koji će biti opisan u ovom radu još jedan korak naprijed prema ostvarenju tog cilja. Na kraju rada, predstavljena je moguća primjena predstavljenog modela u formi programskog rješenja koje prima pjesmu zapisanu u *MIDI* formatu te odabranim vremenskim intervalima pridružuje odgovarajuću oznaku akorda.

1.3 Prijašnji radovi

U zadnjih dvadesetak godina predstavljeni problem se intenzivno proučavao.

Jedan od najznačajnijih i najcitiranijih radova na tu temu je rad Fujishime iz 1999. godine (vidi [14]). U tom radu predstavljen je sustav za automatsko prepoznavanje akorda koji koristi diskretnu Furierovu transformaciju da bi iz ulaznog signala generirao takozvani *Pitch Class Profile* ili *Chromatogram* nad kojim se primijenjuje *pattern matching* pristup. Taj rad, kao i mnogi drugi na tu temu za generiranje atributa (engl. *features*) koji se koriste u istraživanju koristi neku vrstu direktne obrade ulaznog signala (vidi [19] ili [16]).

U radu autora Sheha i Ellisa predstavljena je tehnika prepoznavanja akorda koja koristi *Hidden Markov Models* pristup (vidi [35]). Taj rad nam je zanimljiv iz razloga jer opisani model pokušava obuhvatiti širok spektar mogućih oznaka akorda te se tako postići robusnost na rubne slučajeve. To je nešto na što smo se i mi pokušali koncentrirati u našem istraživanju, budući da je upravo to još uvijek glavna mana većine sustava za automatsko prepoznavanje. Također, korištenjem *Hidden Markov Models* pristupa pokušava se informacija o kontekstu (prijašnjim akordima) ugraditi u sam model što je osnovna ideja našeg pristupa, iako realizirana na drugačiji način što će biti opisano u nastavku. Sličan pristup koristili su i autori Lee i Slaney u svom radu, koji je nastao kao nastavak prije navedenog rada (vidi [20]. U tom radu se kao skup podataka koriste pjesme u *MIDI* formatu, što je karakteristika i našeg pristupa. Ipak, taj rad karakterizira relativno mali broj promatranih klasa akorda pa su samim time dobiveni rezultati neusporedivi s našim pristupom. Nadalje, Cheng i suradnici su u svom radu iz 2008. godine (vidi [9]) pokušali kontekstnu informaciju proširiti korištenjem *N-gram* modela . Taj rad, iako ne postiže očekivanu točnost na testnim primjerima, uvodi pokušaje generiranja novih atributa s ciljem proširivanja informacije o kontekstu.

Rad najuže vezan uz naše istraživanje je *BREVE: an HMPerceptron-Based Chord Recognition System* autora Radicionija i Esposita (vidi [33]) iz 2010. godine. U tom radu je za kreiranje modela za prepoznavanje akorda korišten isti početni skupa podataka kao i u našem radu. Zato su rezultati predstavljeni u tom radu pogodni za usporedbu s našim pristupom. U tom radu, problem prepoznavanja akorda je predstavljen kao *Sequential Learning Problem* te je za učenje modela korišten *CarpeDiem* algoritam (vidi [12]), poboljšanje Viterbijevog algoritma, kako bi sekvencijalna kontekstna situacija bila sačuvana.

Najviša dobivena točnost korištenjem tog algoritma (kako je navedeno u originalnom radu) je 80.06% (10-erostruka unakrsna validacija na svim primjerima). Za razliku od spomenutog rada, u našem radu je korišten drugačiji pristup: kontekstna informacija sačuvana je generiranjem novih atributa, što će biti opisano u nastavku rada.

Poglavlje 2

Podaci

U ovom poglavlju bit će opisan skup podataka korišten za treniranje modela, način razdvajanja na skup za treniranje i testiranje te kako su se originalni podaci koristili za dobivanje dodatnih informacija o samom skupu, što je pomoglo u generiranju novih atributa.

2.1 Osnovni skup podataka

Osnovni skup podataka korišten u ovom radu preuzet je iz baze podataka namijenjenih za rješavanje problema vezanih uz strojno učenje *UCI Machine Learning Repository* (vidi [11]).

Skup se sastoji od 60 korala[1]njemačkog skladatelja Johanna Sebastiana Bacha[2]zapisanih u *MIDI* datotekama iz kojih su izvučeni tonovi koji se pojavljuju u točno određenim vremenskim intervalima. Kasnije u radu ćemo se često na te vremenske intervale referencirati kao događaje (*evente*). Johann Sebastian Bach jedan je od najznačajnijih i najprepoznatljivijih skladatelja svih vremena kojeg, u skladu s vremenom u kojem je djelovao, karakterizira naglašena polifonija, a samim time i česte promjene akorda i njihove varijacije što je izrazito prikladno za problem kojim se bavi ovaj projekt. Podaci se sastoje od 5665 različitih primjeraka i 17 različitih atributa(Tablica 2.1).

Kao što je opisano u Poglavlju 1.2, problem prepoznavanja akorda može se protumačiti kao klasifikacijski problem, gdje se kao klasa koristi oznaka akorda, koja nam je dana kao zadnji atribut u pripadnoj tablici (vidi Tablica 2.1).

[2]Koral označava vrstu melodije koja se pjeva u crkvenim slavljima karakteristična za razdoblje baroka (vidi [28]).

[2]Johann Sebastian Bach (1685. - 1750.), njemački skladatelj (vidi [23])

Tablica 2.1: Atributi u početnom skupu podataka

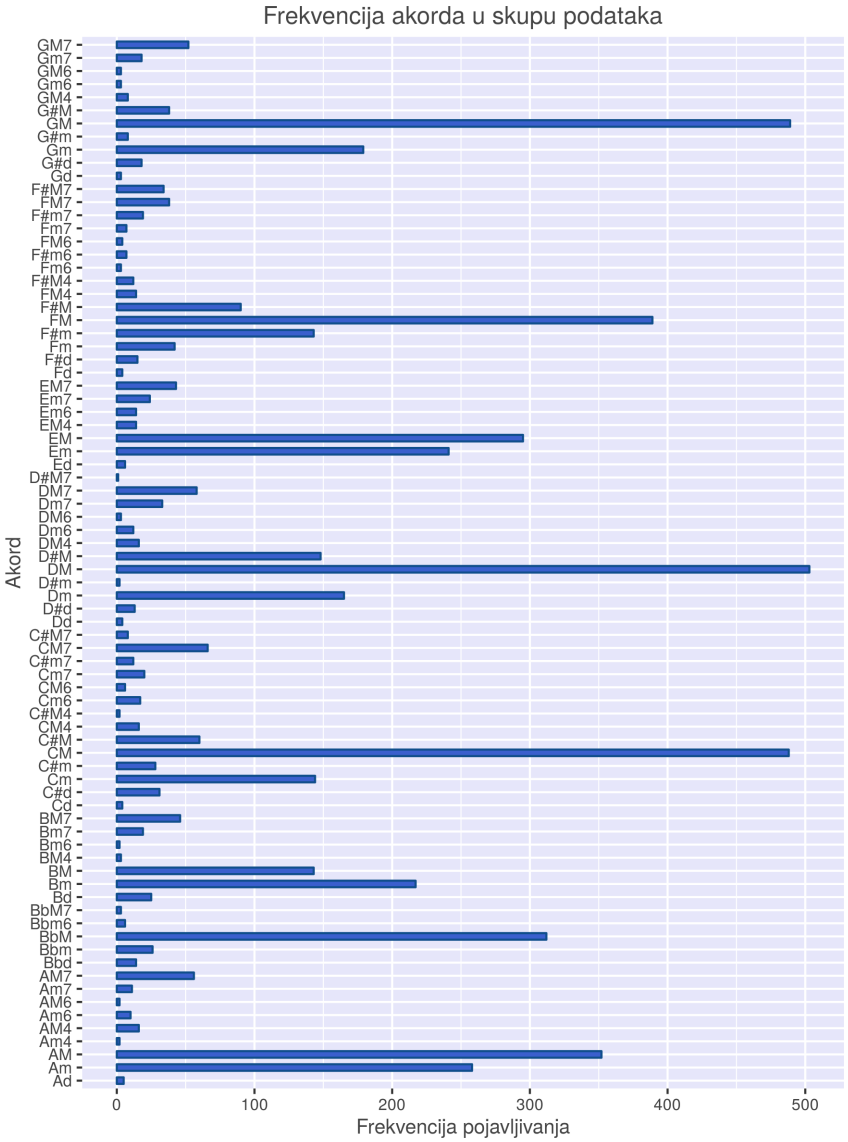
ATRIBUT	OPIS
1	identifikacijski broj (ID) korala
2	redni broj događaja u promatranom koralu
3-14	YES/NO vrijednosti za svaki od 12 tonova (C, C♯, D, D♯, E, F, F♯, G, G♯, A, Bb i B); vrijednost YES predstavlja da se odgovarajući ton (neovisno o oktavi) pojavljuje u odgovarajućem događaju, dok NO znači da se ne pojavljuje u promatranom događaju
15	najdublji ton koji se pojavljuje u događaju (bas nota)
16	cijeli brojevi od 1 do 5 koji označavaju naglašenost primjerka (1 - najmanje, 5 - najviše)
17	oznaka akorda

2.2 Redukcija skupa podataka

Originalni skup podataka sadrži 102 klase, tj. 102 oznake akorda. Međutim, frekvencija klasa jako varira: nekoliko klasa se pojavljuje s jako velikom frekvencijom, a nekoliko klasa s vrlo malom, svega jedan do dva primjerka (vidi Tablica 2.2).

To neslaganje među podacima možemo ublažiti na sljedeći način: ako se za akorde koji se izrazito rijetko pojavljuju (ne samo u ovom skupu podataka, nego i u praksi) kao zamjena uzme neka od srodnih oznaka akorda, čime se ne gubi mogućnost interpretacije dobivenih rezultata, moguće je reducirati broj klasa. Koristeći navedeni princip, u pripadnom skupu podataka smo akorde s oznakama *d7*, *d6* i *d4*[3](na svim tonovima na kojima se pojavljuju, dakle *Cd4*, *Ad7*, itd.) zamijenili s oznakom *d*, gdje je *d* oznaka za smanjeni akord, (engl. *diminished*). Time smo broj klasa s 102 reducirali na 80, a nismo značajnije utjecali na informaciju u skupu podataka. Naime, promijenili smo oznaku akorda na 52 mjesta što je 0.9% skupa podataka što možemo smatrati zanemarivim. Slika 2.1 prikazuje frekvenciju pojavljivanja pojedinog akorda nakon redukcije početnog skupa podataka.

[3]Oznake *d7*, *d6* i *d4* označavaju smanjene akorde s dodatnim rezonirajućim tonovima. Na primjer *d7*, osim smanjenog akorda, sadrži i smanjeni sedmi stupanj (interval *smanjene septime* na početnom tonu). Za više informacija vidi, na primjer [18].



Slika 2.1: Frekvencija pojavljivanja pojedinog akorda nakon redukcije početnog skupa podataka

Tablica 2.2: Frekvencije pojavljivanja oznaka akorda u originalnom skupu podataka

	M	*M4*	*M6*	*M7*	*m*	*m4*	*m6*	*m7*	*d*	*d6*	*d7*
C	488	16	6	66	144	-	17	20	-	2	2
C♯	60	2	-	8	28	-	-	12	12	2	17
D	503	16	3	58	165	-	12	33	-	-	4
D♯	148	-	-	1	2	-	-	-	8	1	4
E	295	14	-	43	241	-	14	24	6	-	-
F	389	14	4	38	42	-	3	7	3	-	1
F♯	90	12	-	34	143	-	7	19	14	-	1
G	489	8	3	52	179	-	3	18	3	-	-
G♯	38	-	-	-	8	-	-	-	12	-	6
A	352	16	2	56	258	2	10	11	5	-	-
B♭	312	-	-	3	26	-	2	-	10	-	4
B	143	3	46	217	-	2	19	17	-	-	8

2.3 Prilagodba početnih atributa

Originalni skup podataka sastoji se od 17 atributa koji su detaljnije opisani u prikazanoj tablici (vidi Tablica 2.1). U radu se, od izvornih atributa za klasifikaciju koristi njih 15 i to oni koji su u prikazanoj tablici označeni rednim brojevima od 3 do 17, pri čemu zadnji atribut, oznaka akorda, koristi kao atribut klase prilikom klasifikacije.

Atribut pod rednim brojem 2, redni broj događaja u koralu, ne koristi se za klasifikaciju jer ne daje dovoljno općenitu informaciju o samom akordu kojeg opisuje. Akord je ovisan o kontekstu u kojem se pojavljuje, ali ne i o relativnom položaju u pjesmi. Kad bi se taj atribut koristio za klasifikaciju, metoda bi mogla naučiti prepoznavati akorde samo preko pozicije u pjesmi, što nije karakteristika problema kojeg promatramo. No, taj atribut će imati vrlo važnu ulogu jer će nam dati mogućnost generiranja dodatnih atributa koje ćemo koristiti u različitim pristupima problemu koji su opisani u sljedećem poglavlju (vidi Poglavlje 3).

Za daljnje dodavanje atributa potrebna nam je definicija *tonaliteta* ili *ključa* pjesme. I ovu definiciju ćemo izreći samo okvirno jer njezino obrazloženje kao i sve ostale formalne definicije teorije glazbe nadilaze potrebe ovog rada (pozivamo čitatelja da za sve detalje pogleda u [34]). Tonalitet ili ključ glazbenog djela definira se kao grupa tonova (*ljestvica*) nad kojom je određeno glazbeno djelo nastalo. Tonalitetu se daje ime s obzirom na prvi

ton koji se pojavljuje u ljestvici (prvi stupanj koji se naziva *tonika*). Za akorde koji pripadaju istom koralu (imaju istu vrijednost prvog atributa, ID korala) vrijedi da su karakteristični za određeni tonalitet i to upravo za tonalitet danog korala. Iz tog razloga, umjesto informacije o identifikacijskom kodu korala koristimo informaciju o tonalitetu tog glazbenog dijela. Na taj način je sačuvana informacija iz skupa podataka, a omogućena je generalizacija te informacije za nove primjere. Također, sam kontekst u kojem se promatrani primjerak pojavljuje je time bolje opisan. U našem slučaju, zbog nedostatka informacija o originalnom nazivu, originalnog notnog zapisa ili pripadne *MIDI* datoteke, informacija o tonalitetu procjenjena je iz ostalih atributa, primarno oznaka akorda koji se pojavljuju u pojedinom koralu. U praksi je ta informacija poznata. Naime, informacija o tonalitetu uvijek se može iščitati iz notnog zapisa same skladbe. Također, ako koristimo pjesme u *MIDI* formatu, često je tonalitet već zapisan u datoteci kao standardna *MIDI key signature* poruka. Vrijedi napomenuti da smo zbog jednostavnosti modela u obzir uzimali samo durski tonalitet. Preciznije, ako je pjesma originalno u molskom tonalitetu, za vrijednost atributa uzeli smo paralelni[4]durski tonalitet.

2.4 Opis i priprema podataka za učenje

Kao što je opisano u Poglavlju 1.1 pojedini primjerak iz skupa podataka (jedan događaj) nema smisla promatrati zasebno, jer je u uskoj vezi s ostalim primjercima iste pjesme. Kako bi se osiguralo da kontekst i veza prema primjercima iste pjesme svakog primjerka budu sačuvani, kao osnovnu jedinicu prilikom podjele podataka na skup za treniranje i testiranje nismo koristili jedan događaj, nego cijelu pjesmu. Kao što smo naveli u Poglavlju 2.1, originalni skup podataka se sastoji od 60 pjesama. Od tih pjesama, na slučajan način odabrali smo njih 44 (73.3%) koje predstavljaju skup za treniranje. Preostalih 16 pjesama smo koristili kao skup za testiranje. Tako se, u ovoj fazi, naš skup za treniranje sastoji od 4360 primjera, a skup za testiranje od 1305 primjera. Budući da se, s odabirom pjesme kao osnovne jedinice prilikom podjele početnog skupa podataka, broj promatranih jedinica svodi na tek 60, odlučili smo da nećemo generirati validacijski skup kojeg bismo koristili za održivanje najboljih parametara modela kako bismo izbjegli dodatno smanjivanje skupa za treniranje. Umjesto toga, odabir parametara provodili smo na skupu za treniranje korištenjem metode unakrsne validacije.

[4]Kažemo da su durska i molska ljestvica paralelne ako imaju identične predznake (povisilice ili snizilice). Nadalje, početni ton (*tonika*) paralelnog mola je za interval *malu tercu* niža od početnog tona durskog tonaliteta (detalji u [34])

Zbog načina na koji su akordi građeni u teoriji glazbe (vidi [26] i [30]), moguće je početni skup podataka dodatno proširiti.

Kako je opisano u Poglavlju 1.1 svaki akord se sastoji od određenog broja tonova. Primjerice, pojava tonova C, E i G, u odgovarajućem kontekstu, može simbolizrati akord C dura (oznaka akorda CM). No, ukoliko se tonovi C, E, G povise za pola tona, dobije se kombinacija tonova $C\sharp$, $E\sharp$ (odnosno F[5]), $G\sharp$ koja simbolizira $C\sharp$ dur (oznaka $C\sharp M$. Na opisani način, moguće je od jednog akorda povisivanjam tonova za pola tona dobiti nove akorde, a zadržati informaciju o početnom kontekstu opisanim danim primjerom u skupu podataka. Kako je ukupan broj tonova koje promatramo 12, to znači da se od jednog događaja u skupu podataka može generirati još 11 novih događaja[6].

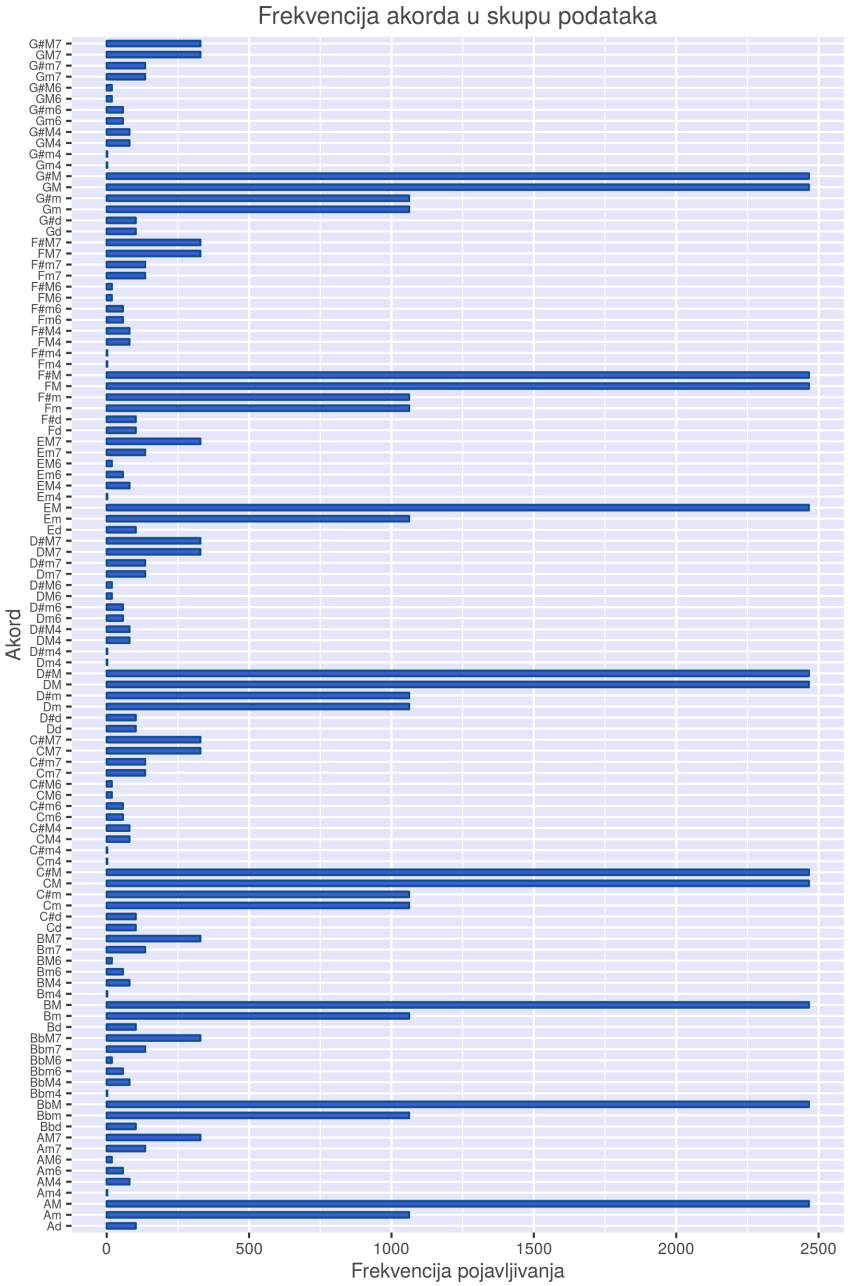
Koristeći navedeni pristup početni skup podataka za treniranje proširili smo 12 puta, odnosno svaku pjesmu u skupu za treniranje prebacili smo u 12 različitih tonaliteta. Samim time, postigli smo da je informacija koja je zapisana u nekoj određenoj pjesmi neovisna o tonovima koji se pojavljuju u njoj, nego je naglašen uzorak po kojem se akordi mijenjaju što nam daje puno općenitiju informaciju. Ovim postupkom kontekst zapisan u skupu podataka prenosimo na sve tonove čime povećavamo opseg moguće primjene generiranog modela. Naime, istu informaciju o nekom akordu možemo koristiti na svakom od promatranih tonova.

Dodatna prednost ovog pristupa je što je na taj način osigurano da se u skupu za treniranje pojavljuju sve promatrane klase. Štoviše, svaka klasa će se pojaviti minimalno 12 puta.

Kao što je već navedeno, opisanim proširivanjem dobili smo i neke nove primjerke klasa zbog čega je u konačnici broj klasa u novodobivenom skupu podataka za treniranje 108. Skup za treniranje sada sadrži 51012 primjera te će u tom obliku biti korišten za treniranje modela. Slika 2.2 prikazuje frekvencije pojavljivanja pojedinog akorda u proširenom skupu za treniranje.

[5] $E\sharp$ i F predstavalju isti ton po zvučnosti na kojeg se, ovisno o ljestvici, drugačije referencira. Slično je i sa tonovima $C\sharp$ i $D\flat$ i drugima koji se u nekom trenutku mogu ponašati kao povišeni ($\sharp$), au drugom kao sniženi ($\flat$) tonovi. U ovom radu se, zbog jednostavnosti modela, neovisno o tonalitetu, uvijek referenciramo na povišeni ton (za sve detalje vidi [30]).

[6] Takav postupak se u strojnom učenju obično naziva *data augmentation* (za detalje vidi [37]).



Slika 2.2: Frekvencija pojavljivanja pojedinog akorda u skupu za treniranje nakon proširivanja

Poglavlje 3

Plan rada

U jednom od već navedenih prijašnjih radova na temu prepoznavanja glazbenih akorda od strane autora Radicionija i Espositoa *BREVE: an HMPerceptron-Based Chord Recognition System* (vidi [33]) navedeno je da standardni pristupi kojima se rješavaju klasifikacijski algoritmi poput stabla odlučivanja ili *Naive Bayes* algoritma ne daju zadovoljavajuće rezultate jer ne mogu u svoja predviđanja uključiti informaciju o kontekstu. Upravo iz tog razloga, kako je već navedeno u Poglavlju 1.3, u tom radu i u ostalim radovima koji se bave varijantama tog problema koriste se *Supervised Sequential Learning* pristupi koji se baziraju na korištenju *Hidden Markov Models* ili *N-gram* modela te varijantama *Viterbijevog* algoritma (vidi [12]).

U ovom radu odlučili smo koristiti drugačiji pristup te smo za klasifikaciju koristili algoritam *Support Vector Machines* (vidi Poglavlje 3.6) s time da smo informaciju o kontekstu pokušali ugraditi u sam skup podataka generiranjem dodatnih atributa.

Rad smo podijelili u 4 faze:

1. *Faza 1*: Klasifikacija na polaznom skupu podataka
2. *Faza 2*: Klasifikacija na skupu podataka s proširenim skupom atributa
3. *Faza 3*: Klasifikacija koja simulira slijedni pristup
4. *Faza 4*: Klasifikacija koja koristi distribuiranu reprezentaciju zasnovanu na *word2vec* algoritmu

Svaka od navedenih faza bit će detaljnije opisana u nastavku rada.

3.1 Faza 1: Klasifikacija na polaznom skupu podataka

U prvoj fazi, analiziramo ponašanje *SVM* algoritma na polaznom skupu podataka, odnosno, na skupu koji uključuje atribute opisane u Poglavlju 2 s dodatnom informacijom o tonalitetu (vidi Poglavlje 2.3). Koristimo podjelu na skup za treniranje i skup za testiranje opisanu u Poglavlju 2.4.

Cilj prve faze je analiza rezultata dobivenih koristeći samo osnovni skup atributa kako bismo mogli ocijeniti eventualna poboljšanja modela u kasnijim fazama.

3.2 Faza 2: Klasifikacija na skupu podataka s proširenim skupom atributa

U drugoj fazi, polaznom skupu podataka dodali smo 3 nova atributa koje smo generirali na temelju informacija dobivenih iz poznatih vrijednosti atributa s ciljem proširivanja informacija o kontekstu u kojem se neki akord pojavljuje. Cilj ove faze je nadopuniti model informacijama koje već inicijalno postoje u skupu podataka, no nisu eksplicitno definirane kao atribut. Novogenerirani atributi su: prethodna bas nota, mala terca i velika terca.

3.2.1 Generiranje prethodne bas note

Prvi atribut koji smo generirali s ciljem proširivanja konteksta je prethodna bas nota. Pri generiranju tog atributa uvelike nam je pomogao atribut pod rednim brojem 2 u početnoj tablici (vidi Tablica 2.1) koji označava redni broj događaja. Svakom događaju iz skupa podataka za testiranje smo, koristeći tu informaciju, pridružili informaciju o bas noti (atribut pod rednim brojem 15 u Tablici 2.1) prethodnog događaja. Ako je promatrani primjerak prvi događaj u nekoj pjesmi, za prethodnu bas notu uzimali smo osnovni ton tonaliteta (*toniku*). Takav pristup smatramo izrazito pogodnim za ovu vrstu problema jer su prijelazi između dva akorda u glazbi često realizirani upravo u bas liniji.

3.2.2 Generiranje male i velike terce

S ciljem proširivanja konteksta, u ovoj fazi uveli smo još dva dodatna atributa, podatak postoje li u promatranom događaju dva tona koja čine inter-

C	C♯	D	D♯	E	F	F♯	G	G♯	A	B♭	B
+	-	+	-	+	-	-	-	-	+	-	-

Slika 3.1: Primjer pojavljivanja velike i male terce

val[1] *veliku tercu*, odnosno *malu tercu*.

Velika terca definira se kao interval koji se sastoji od dva cijela stepena, tj. 4 polustepena[2].

Mala terca definira se kao interval koji se sastoji od jednog cijelog i jednog polustepena, tj. 3 polustepena.

Promatrajući naš vektor pojavljivanja tonova (vidi Tablica 2.1, atributi 3-14), velika terca će se pojaviti ako postoje dva tona koja su udaljena za točno 3 mjesta. Slika 3.1 pokazuje primjer događaja u kojem se pojavljuju i velika i mala terca. U tom događaju pojavljuju se tonovi C, D, E i A. C i E su na Slici 3.1 udaljeni za točno 4 mjesta (4 polustepena) te čine interval veliku tercu (još jednom napominjemo da ne promatramo u kojoj oktavi se nalaze tonovi, već nam je bitno samo da se pojavljuju). Nadalje, tonovi A i C su na Slici 3.1 udaljeni za točno 3 mjesta (gledamo pomak modulo 12) te čine interval malu tercu.

Podatak o velikoj i maloj terci uvelike može pridonijeti raspoznavanju konteksta i to u slučaju raspoznavanja razlike između durskog i molskog akorda nad istim tonom zbog načina na koji su durski i molski akord građeni. Detaljnije informacije o građi akorda mogu se naći u [26], [30], [34].

3.3 Faza 3: Klasifikacija koja simulira slijedni pristup

U trećoj fazi simulirali smo slijedni (engl. *sequential*) pristup problemu. Preciznije, za generiranje dodatnih atributa koristili smo i informaciju o oznaci akorda, tj. atribut klase u našem klasifikacijskom problemu. Za početak, u svrhu dodatnog proširivanja konteksta promatrali smo prethodna pojavljivanja akorda u istoj pjesmi. Time smo uveli 3 nova atributa koja će biti opisana u nastavku.

[1] Pojam interval u teoriji glazbe označava udaljenost između dva tona.

[2] Jedan polustepen je razmak između dva susjedna tona, na primjer C-$C\sharp$, $G\sharp$-A i slično.

3.3.1 Promatranje prethodnih pojava akorda

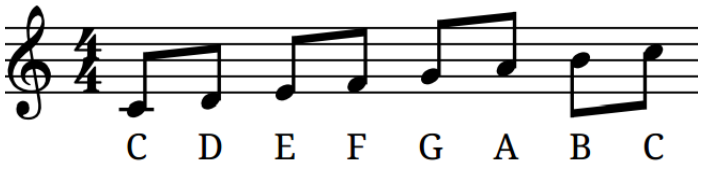
Prvi novopromatrani atribut je oznaka prethodnog akorda u istoj pjesmi. Pri generiranju tog atributa smo također koristili atribut pod rednim brojem 2 u polaznoj tablici (vidi Tablica 2.1), kao i kod generiranja prethodne bas note (vidi Poglavlje 3.2.1) koji označava redni broj događaja. Svaki primjerak iz skupa podataka za treniranje je tako dobio informaciju koji akord je prethodio tom događaju (u istoj pjesmi). Ako je promatrani primjerak prvi događaj u nekoj pjesmi, kao osnovni akord uzimali smo durski akord na osnovnom tonu pripadnog tonaliteta (*tonici*). Nadalje, promatrali smo koji od akorda se najčešće pojavljuje u predhodnih 10 događaja (ili manje ako gledamo početak pjesma).

3.3.2 Dijatonička funkcija akorda i njezina primjena na skup podataka

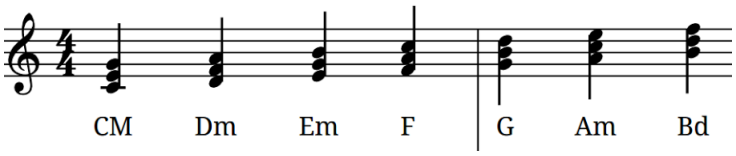
U teoriji glazbe, vežnu ulogu ima *dijatonička funkcija* akorda (u literaturi se još naziva i *harmonijska funkcija*, *okolina akorda* ili samo *funkcija*). Ukratko, taj pojam označava veze pojedinih akorda s obzirom na početni ton ljestvice (*toniku*). Nadalje, koncept dijatoničke funkcije govori o međusobnom slaganju i važnosti pojedinih akorda u danom tonalitetu. Detaljniji opis pojma dijatoničke funkcije uvelike izlazi iz okvira ovog rada i za sam problem koji proučavamo je opisana okvirna definicija dovoljna. No, pozivamo znatiželjnog čitatelja da za detaljnije razumijevanje prouči [27].

Uvođenjem pojma dijatoničke funkcije u glazbi, postiže se da za svaki tonalitet možemo odrediti skupinu akorda za koje smatramo da se slažu u nekom smislu, tj. da na neki način odgovaraju danom tonalitetu. U Primjeru 3 opisano je na koji način se određuju odgovarajući akordi s obzirom na dijatoničku funkciju promatrajući *C dur* ljestvicu. Iz primjera zaključujemo da je skupina akorda usko vezana uz tonalitet *C dur* upravo skup koji se sastoji od akorda *CM*, *Dm*, *Em*, *FM*, *GM*, *Am* te *Bb*.

U skladu s prethodnim razmatranjima, odlučili smo našem skupu podataka dodati još jedan atribut. Promatrali smo koji udio od prethodnih 10 akorda s obzirom na promatrani događaj pripada skupu bliskih akorda definiranih na način kako je opisano ranije, u Primjeru 3. Drugim riječima, gledali smo udio prethodnih akorda koje smatramo očekivanima za dani tonalitet. Treba napomenuti da smo u samim skupovima bliskih akorda napravili jednu promjenu. Akord s oznakom *d*, tj. smanjeni akord (*diminished*) u praksi se pojavljuje relativno rijetko, a u danom tonalitetu se često koristi durski akord na trećem stupnju ljestvice. Koristeći to saznanje (posebno se to vidi u modernoj zapadnjačkoj glazbi), svakom skupu dobivenom na gornje prikazani



Slika 3.2: *C dur* ljestvica



Slika 3.3: Akordi karateristični za *C dur* ljestvicu

način, smanjeni akord na sedmom stupnju zamijenili smo durskim akordom na trećem stupnju. Na taj način, ponovno na primjeru *C dur* ljestvice dobivamo sljedeći skup: *CM*, *Dm*, *Em*, *FM*, *GM*, *Am* te *E*. Analogno se prikažu skupovi za ostale tonalitete koristeći translaciju za pola tona kako je već opisanu na primjeru proširivanja skupa podataka (vidi 2.4).

Primjer 3. *Promotrimo C dur ljestvicu. Ona se sastoji od tonova C, D, E, F, G, A, B, C (vidi Slika 3.2). Izgradimo trozvuke (akorde koji se sastoje od 3 tona) na svih 7 stupnjeva ljestvice od tonova koji se pojavljuju u ljestvici (za detalje vidi [27]). Dobivenim akordima pridružujemo oznake kao na Slici 3.3. To su upravo akordi koje smatramo da su karakteristični za C dur ljestvicu u smislu dijatoničke funkcije.*

3.4 Slijedni pristup - algoritam

Uočimo da sva 3 novogenerirana atributa koriste informacije o prethodnim akordima, koji su zapravo atribut klase prilikom klasifikacije. To ne predstavlja problem kod proširenja skupa za treniranje jer su atributi klase već poznati pošto se naš skup podataka sastoji od već labeliranih primjera. Međutim, u skupu za testiranje, kao i u stvarnim situacijama, tu informaciju nemamo.

Kao što je već navedeno u uvodnom poglavlju (vidi Poglavlje 1), jedno od mogućih primjena rezultata ovog rada je program kojim bi se za danu pjesmu kao ulaz prepoznavali akordi s obzirom na točno određeni vremenski interval (moguće i u stvarnom vremenu). U takvim primjenama će svakom događaju koji je prethodio trenutno promatranom događaju već biti pridružena odgovarajuća oznaka akorda, budući da će se pjesma obrađivati sekvencijalno. To

znači da tu infomaciju možemo koristiti kod klasifikacije sljedećeg primjera iz iste pjesme. To nam daje mogućnost da simuliramo slijedni pristup problemu, tj. da koristimo informacije o prethodnim akordima na način da nove atribute generiramo "u hodu", prilikom klasifikacija i na taj način proširujemo kontekst. Osnovna ideja prikazana je pseudokodom (vidi Algoritam 1).

Uočimo jednu manu ovog pristupa. Kako se prilikom generiranja atributa koristi informacija o prethodno klasificiranim primjerima, eventualna greška se ugrađuje u nove primjere. Iz tog razloga bi promatranje prevelikog broja primjera iz prošlosti moglo negativno utjecati na samu klasifikaciju. Za broj promatranih primjera "u prošlost" najboljom se pokazala konstanta 10, tj. promatranjem 10 prethodnih primjera postizali su se najbolji rezultati.

Algorithm 1 Pseudokod algoritma koji simulira slijedni pristup

```
1: classifier = svmClassifier
2: k = 10
3: previousChords = empty
4: foreach event in song
5:     generateFeatures(event, previousChords, k)
6:     prediction = classify(event, classifier)
7:     add(prediction, previousChords)
8: end foreach
```

3.5 Faza 4: Klasifikacija koja koristi distribuiranu reprezentaciju zasnovanu na *word2vec* algoritmu

U ovoj fazi, model smo pokušali poboljšati uvođenjem dodatne informacije o kontekstu. Naime, u prethodnoj fazi, informacija o događajima koji su prethodili promatranom događaju je uključivala samo informaciju o prethodnom akordu i akordu koji se do tada najčešće pojavljivao. Uočimo da se takvim pristupom ne obuhvaća informacija o potpunoj okolini akorda koja uključuje sve akorde koji prethode, ali i one koje slijede danom akordu.

3.5.1 Word2Vec

Budući da kontekst u ovom problemu ima jako veliki utjecaj na učenje modela (vidi Poglavlje 1.1), odlučili smo na naš problem primijeniti jednu od metoda koja se koristi u domenama primjene strojnog učenja koji zahtjevaju očuvanje

konteksta. Jedan od problema takvog tipa je procesiranje jezika (*Natural Language Processing*), a neki od primjera su prevođenje iz jednog jezika na drugi, predviđanje sljedeće riječi u rečenici, ali i problem reprezentacije riječi nekog jezika vektorom fiksne duljine. Upravo je reprezentacija riječi vektorom fiksne duljine ideja koju smo primijenili na naš problem. Formalno, vektorska reprezentacija riječi, poznata pod nazivom *word embedding* je funkcija

$$W : \Sigma \to \mathbb{R}^n$$

koja preslikava riječ nekog jezika Σ u n-dimenzionalni vektor. Na primjer, ako za jezik uzmemo skup svih riječi engleskog jezika, naša funkcija W može na primjer riječi "cat" pridružiti neki 5-erodimenzionalni vektor: $W(\text{"}cat\text{"}) = (0.2, -0.4, 0.7, 0.5, -0.2)$ (za detalje, vidi [29]).

Najjednostavniji primjer je reprezentacija riječi nekog jezika Σ pomoću bit-vektora, odnosno za jezik koji se sastoji od n riječi, reprezentacija riječi $r_i, i = 1, .., n$ je jedinični vektor $e^i \in \mathbb{R}_n$ sa svojstvom da su svi elementi vektora e_i jednaki 0, osim i-tog koji je jedan. Preciznije, $e^i_j = \delta_{ij}$ gdje je δ_{ij} Kroneckerov simbol. Primjerice, jedna moguća reprezentacija riječi jezika $\Sigma = \{\text{"}mačka\text{"}, \text{"}pas\text{"}, \text{"}kralj\text{"}, \text{"}kraljica\text{"}\}$ prikazana je u Primjeru 4.

Primjer 4. *Promotrimo jezik* $\Sigma = \{\text{"}mačka\text{"}, \text{"}pas\text{"}, \text{"}kralj\text{"}, \text{"}kraljica\text{"}\}$. *Jedna reprezentacija riječi tog jezika pomoću bit-vektora je sljedeća funkcija:*

$$W : \Sigma \to \mathbb{R}^n$$

definirana na sljedeći način:

$$W(\text{"}mačka\text{"}) = (1, 0, 0, 0)^T$$
$$W(\text{"}pas\text{"}) = (0, 1, 0, 0)^T$$
$$W(\text{"}kralj\text{"}) = (0, 0, 1, 0)^T$$
$$W(\text{"}kraljica\text{"}) = (0, 0, 0, 1)^T$$

Međutim, ovakva reprezentacija daje jedino informaciju o egzistenciji pojedine riječi, ali ne i o kontekstu. Za modeliranje složenijih reprezentacija koristi se učenje preslikavanja u gust vektor uz pomoć neuronskih mreža (vidi [29]). Glavna ideja uvođenja ovakvog preslikavanja je da reprezentacije riječi u vektor imaju smisla i u modeliranju konteksta. Preciznije, ideja je da riječi koje često u rečenicama dolaze zajedno budu preslikane "blizu" jedna drugoj, a riječi koje se zajedno rijetko pojavljuju budu što udaljenije (u smislu n-dimenzionalnog prostora.)

Jedna od najpoznatijih metoda kojom se postiže takvo preslikavanje je poznata pod nazivom *word2vec* autora Mikolova i suradnika (vidi [25]). Model za *word2vec* pristup je dvoslojna neuronska mreža trenirana tako da za

ulaznu riječ predviđa njezin kontekst, odnosno riječi koje ulaze u kontekst dane riječi. Model ove arhitekture se naziva *skip-gram* (vidi [25]).

Iz modela istreniranog na opisani način, vektorsku reprezentaciju svake riječi moguće je dobiti aktiviranjem težina u srednjem (skrivenom) sloju. Zbog toga broj slojeva u srednjem sloju odgovara dimenziji vektora reprezentacije riječi.

Ono što se pokazalo značajno u ovakvoj reprezentaciji je smisleno grupiranje riječi u prostoru: pokazalo se da su slične i povezane riječi međusobno "blizu". Štoviše, pokazalo se kako razlike parova vektora nekih sinonima konstantne. Primjerice:

$$W(\text{"}\check{z}ena\text{"}) - W(\text{"}muškarac\text{"}) \simeq W(\text{"}kraljica\text{"}) - W(\text{"}kralj\text{"})$$
$$W(\text{"}Njemačka\text{"}) - W(\text{"}Berlin\text{"}) \simeq W(\text{"}Francuska\text{"}) - W(\text{"}Pariz\text{"})$$
$$W(\text{"}toplo\text{"}) - W(\text{"}toplije\text{"}) \simeq W(\text{"}hladno\text{"}) - W(\text{"}hladnije\text{"})$$

Za još primjera čitatelju preporučujemo [36].

3.5.2 Korištenje *word2vec* modela s ciljem proširivanja kontektsta

Word2vec pristup odlučili smo primijeniti na problem prepoznavanja akorda. Za jedan od prijašnjih radova koji se bavi učenjem vektorske reprezentacije akorda vidi [22].

U prethodnoj fazi, od tri novouvedena atributa, dva su podatak o prethodnom akordu i informacija o najčešće prisutnom akordu. Međutim, budući da se radi o kategoričkim, nenumeričkim vrijednostima (108 mogućih akorda), svaki akord je, preslikan u jedinični vektor duljine 108 te ga klasifikacijski algoritam koristi u tom obliku. Odnosno, umjesto jednog atributa *prethodni akord* zapravo se radi o informaciji sa 107 vrijednosti koje su jednake 0 i točno jednom vrijednosti 1.

Ekvivalentno problemima procesiranja jezika, ovakva reprezentacija akorda ne daje nikakvu dodatnu informaciju o kontekstu akorda. Ovakvoj reprezentaciji je svejedno je li redoslijed akorda G, D, A ili A, G, D što može imati veliki utjecaj na uspješnost modela.

Kako bismo dobili vektorsku reprezentaciju akorda koja nosi više informacija, odlučili smo se kreirati model po uzoru na *word2vec* koji će za dani akord predviđati kontekst akorda te, nakon treniranja, omogućiti dobivanje njihovih vektorskih reprezentacija. Prvi korak bio je utvrditi korespodenciju između jezika i glazbe, odnosno odrediti koji *dijelovi* glazbe odgovaraju

kojim *dijelovima* jezika. Korespodenciju smo definirali na sljedeći način:

$$riječ \Leftrightarrow akord$$
$$rečenica \Leftrightarrow pjesma$$

Nakon toga, kreirali smo model ekvivalentan *word2vec* modelu za učenje reprezentacija riječi, uz broj slojeva srednjeg sloja jednak 100. Dimenzija 100 se u našem slučaju pokazala kao najbolji izbor, jer model ne čini presloženim i ne vodi ka pretreniranju.

Za treniranje *word2vec* modela koristili smo iste 44 pjesme od kojih se sastoji skup za treniranje opisan u poglavlju 2.4. Pomoću dobivenog modela reprezentirali smo svaki akord vektorom dimenzije 100. Kako bismo pokazali da dobivena reprezentacija ima smisla u terminu očuvanja konteksta, rezultate smo vizualizirali koristeći *Stohastic Neighbour Embedding*, tj. *t-SNE* (vidi [21]). Zadnji korak bio je modificirati model iz Faze 3 (vidi Poglavlje 3.3) da umjesto dva kategorička atributa *prethodni akord* i *najčešći akord* uzima naučene *word2vec* reprezentacije, dakle dva vektora dimenzije 100.

3.6 Support Vector Machines (SVM)

Support Vector Machines (*SVM*) je klasifikacijski algoritam koji radi na principu pronalaska razdvajajuće hiperravnine koja ima najveću marginu razdvajanja klasa. Preciznije, podaci se promatraju kao p-dimenzionalni vektori te je cilj algoritma pronaći $(p-1)$-dimenzionalnu hiperravninu koja ih razdvaja.

Algoritam se u praksi najčešće koristi nad podacima koji su linearno separabilni (linearno razdvojivi). No, *SVM* se može koristiti i nad linearno neseparabilnim podacima koristeći takozvani *kernel trik*. Sam algoritam je prvi put opisan od strane Vapnika i suradnika početkom devedesetih godina prošlog stoljeća (vidi [7]).

Rad algoritma je detaljnije obrazložen u nastavku. Za eventualne nejasnoće upućujemo čitatelja na knjigu *The Nature of Statistical Learning Theory* Vladimira Vapnika, originalnog autora algoritma (vidi [38]). Također, rad algoritma je opisan i u knjizi *Numerical Recipes: The Art of Scientific Computing (3rd edition)* od strane autora *Pressa* i suradnika (vidi [31]). Navedene knjige su korištene kao osnovna referenca u sljedećim odlomcima.

3.6.1 SVM algoritam

Zamislimo da imamo klasifikacijski problem učenja pod nadzorom (*supervised learning*) sa skupom za treniranje koji se sastoji od m točaka

$$(x_i, y_i), i = 1, ..., m \tag{3.1}$$

gdje je $x_i \in \mathbb{R}^n$ te $y_i \in \{-1, 1\}$ za svaki $i = 1, ..., m$. Uočimo da imamo samo dvije klase, -1 i 1. Cilj je, kao i u svakom klasifikacijskom problemu, pronaći funkciju $f : \mathbb{R}^n \to \{-1, 1\}$ koja vektoru x pridružuje odgovarajuću klasu. Osnova ideja je da odabrana funkcija korektno određuje klasu primjera iz skupa za treniranje, no da ispravno (što je moguće točnije) klasificira i nove, još neviđene, primjere.

Promotrimo najprije slučaj kada su podaci za treniranje linearno separabilni, tj. kada postoji hiperravnina zadana s $a^\tau x + \beta = 0$ takva da za skupove $X_1 = \{x_i : y_i = 1\}$ i $X_2 = \{x_i : y_i = -1\}$ vrijedi da je $a^\tau x + \beta > 0$ za svaki $x \in X_1$ i $a^\tau x + \beta < 0$ za svaki $x \in X_2$ (uočimo da smo s $a^\tau x$ označili skalarni produkt vektora a i x). U tom slučaju možemo odabrati dvije paralelne hiperravnine koje razdvajaju točke koje pripadaju jednoj klasi ($y_i = 1$) odnosno drugoj klasi ($y_i = -1$) na način da njihova udaljenost bude maksimalna. Skup koje određuju te dvije hiperravnine nazivamo *margina*. *Hiperravnina s najvećom marginom razdvajanja* je hiperravnina koja leži točno na polovici između odabranih hiperravnina. Odabrane ravnine mogu se opisati izrazima $a^\tau x + \beta = 1$ te $a^\tau x + \beta = -1$. Njihova udaljenost je $\frac{2}{\|a\|}$. Zaključujemo da, ako želimo pronaći maksimalnu udaljenost između hiperravnina trebamo minimizirati $\|a\|$. Nadalje, svaka točka mora ležati izvan zadane margine što opisujemo izrazima

$$\begin{aligned} &a^\tau x + \beta \geq 1, \text{ ako je } y_i = 1 \\ &a^\tau x + \beta \leq -1, \text{ ako je } y_i = -1 \end{aligned} \tag{3.2}$$

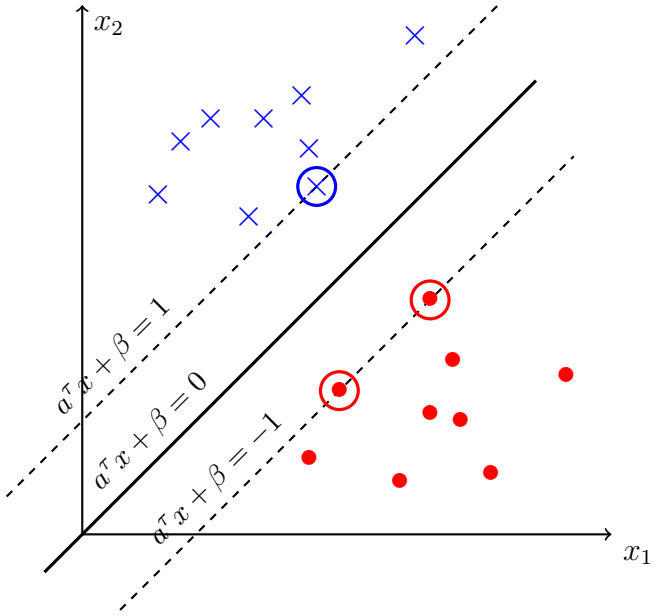
Na taj način dolazimo do sljedećeg optimizacijskog problema:

$$\begin{cases} \|a\| \to \min \\ y_i \left(a^\tau x + \beta\right) \geq 1, \text{ za } i = 1, ..., n \end{cases} \tag{3.3}$$

Dobiveni a i β koji rješavaju taj problem određuju našu klasifikacijsku funkciju

$$f(x) = \text{sgn}(a^\tau x + \beta) \tag{3.4}$$

Vektore za koje je $a^\tau + \beta = \pm 1$ nazivamo *potpornim vektorima* (*support vectors* iz naziva algoritma).



Slika 3.4: Hiperravnina s maksimalnom marginom razdvajanja i potporni vektori

Uočimo da je problem opisan u 3.3 problem konveksnog programiranja koji ima ekvivalentni *dualni* problem. Dualni problem "živi" u prostoru dimenzije m (broj podataka u skupu za treniranje) što čini dimenziju prostora atributa irelevantnom. Za detalje vidi [31].

Da bi se *SVM* algoritam proširio i na slučajeve kad uvjet linearne separabilnosti nije zadovoljen, koristi se funkcija

$$\max\left(0, 1 - y_i(a^\tau x + \beta)\right) \ . \tag{3.5}$$

Ta funkcija je jednaka 0 ako je zadovoljen uvjet da se x nalazi na odgovarajućoj strani margine. Za podatke koji se nalaze na pogrešnoj strani margine, vrijednost funkcije je proporcionalna udaljenosti od margine. U tom slučaju minimiziramo

$$\left[\frac{1}{n}\sum_{i=0}^{n}\max\left(0, 1 - y_i(a^\tau x + \beta)\right)\right] + \lambda\|a\|^2 \tag{3.6}$$

gdje je λ regularizacijski parametar.

Boser, Guyon i Vapnik su u svom radu iz 1992. godine (vidi [7]) predstavili način za generalizaciju SVM algoritma i na nelinearnu klasifikaciju, takozvani *kernel trik*. Ideja algoritma je, formalno, jednaka kao i kod klasičnog *SVM*-a s razlikom da se, umjesto skalarnog produkta, koristi neka

nelinearna funkcija koju nazivamo *kernel* (za detalje o načinu korištenja kernel funkcija i svojstvima često korištenih kernel funkcija vidi [31]). Neke od često korištenih kernel funkcija su:

- *Linearni kernel*
$$K(x_i, x_j) = x_i^\tau x_j$$
- *Polinomijalni kernel*
$$K(x_i, x_j) = (ax_i^\tau x_j + b)^d$$
- *Sigmoidalni kernel*
$$K(x_i, x_j) = \tanh(ax_i^\tau x_j + b)$$
- *RBF (radial basic function) kernel*
$$K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right), \text{ za } \gamma > 0, \text{ ponekad } \gamma = \frac{1}{2\sigma^2}$$

3.6.2 Korištenje SVM algoritma u radu

U našem radu koristili smo varijantu *SVM* algoritma s *RBF* kernel funkcijom. Ta verzija algoritma se često koristi u slučaju skupa podataka koji se sastoji od relativno malog broja atributa u odnosu na broj podataka, što je upravo karakteristika našeg problema. Osim kernel funkcije bilo je potrebno odabrati i odgovarajuće (regularizacijske) parametre te funkcije: penalizacijski parametar C (obrnuto proporcionalan parametru λ iz 3.6) te parametar γ u definiciji *RBF* kernel funkcije. Parametar C kontrolira kompromis (*trade-off*) između točne klasifikacije unutar skupa za treniranje i "glatkoće" granica između klasa. Za fiksan γ i $C \to 0$, dolazi do visoke pristranosti što rezultira pretreniranjem, dok se za $C \to \infty$ pristranost smanjuje. S druge strane, za fiksan C i $\gamma \to 0$ *RBF* funkcija se ponaša kao linearna, čime se smanjuje kompleksnost modela i smanjuje pristranost, dok se za $\gamma \to \infty$ povećava kompleksnost pa time i mogućnost pojave pretreniranja (vidi [17]). Kod treniranja *SVM* modela uobičajeno je fiksirati jedan od parametara γ ili C, a s drugim parametrom kontrolirati kompleksnost modela (vidi [13]). Oba parametra u svakoj od četiri faze odabrana su s obzirom na najbolji rezultat unakrsne validacije na skupu za treniranje.

3.7 Analiza uspješnosti i greške

Kako je već više puta navedeno u ovom radu, problem prepoznavanja akorda je vrlo specifičan, ne samo u smislu kreiranja klaisifikacijskog modela već i u smislu analize uspješnosti, odnosno greške tog modela. U ovom odlomku bit će opisano na koji način smo mjerili uspješnost modela te uspoređivali rezultate u različitim fazama opisanih u prethodnom poglavlju.

3.7.1 Mjera uspješnosti

Kao mjeru uspješnosti promatrali smo točnost modela na skupu za testiranje jer nas je primarno zanimala stvarna točnost predikcije te želimo da robusnost našeg modela na rubne slučajeve bude što bolja.

3.7.2 Distribucija greške

Specifičnost ovog problema leži u tome da nije svaka greška "jednako pogrešna". Primjerice, ako se događaj čija prava oznaka akorda je dominantni septakord na tonu C (oznaka $CM7$) klasificira kao $C\ dur$ akord (oznaka CM), ta klasifikacija nije toliko pogrešna kao klasifikacija istog događaja u, na primjer, $C\sharp dur$ (oznaka $C\sharp M$) ili $F\sharp mol$ (oznaka $F\sharp m$) jer je funkcija akorda u pjesmi očuvana. Štoviše, u praksi često razlike ovoga tipa (CM i $CM7$) ne predstavljaju veliki problem, jer su takve razlike (pogotovo ako se gledaju manji vremenski intervali) teško primjetne ili uvelike ovise o subjektivnom mišljenju autora koji samom akordu pridružuje njegovu oznaku. Međutim, to nije slučaj s navedenim primjerom "jako pogrešne" klasifikacije (CM - $C\sharp M$). Pogreška ovoga tipa obično rezultira disonantnim tonovima (koji su primjetni čak i manje iskusnom uhu). Iz tog razloga odlučili smo analizi greške pristupiti na način da greške podijelimo u nekoliko kategorija. Podjelom greške na kategorije analizirali smo koliko pogrešnih predikcija smatramo izrazito pogrešnima (greška 5. vrste) te koliko pogrešnih predikcija bi se gotovo moglo zanemariti (pogreška 1. vrste). U nastavku dajemo opis svake od promatrane kategorije greške.

1. *Greška 1. vrste: Pogreška u stupnjevima 4, 6 i 7*

   Greška 1. vrste pojavljuje se kad je osnovni dio akorda (početni ton te oznaka za dur ili mol) potpuno jednak, no nedostaje informacija o dodatnom stupnju koji rezonira (na primjer, $CM7$ se klasificira kao CM, $Am6$ kao Am i slično). Greške tog tipa ne smatramo značajnima jer se u praksi, ako se promatra mali vremenski interval, pogreške tog tipa mogu zanemariti, a u nekim slučajevima ih nije niti moguće otkriti.

2. *Greška 2. vrste: Pogreška dur-mol7*

   Greška 2. vrste pojavljuje se kad za akorde koji su označeni oznakom *M*, tj. durske akorde, klasifikator predvidi oznaku *m7* paralelnog mola i obratno. To također ne smatramo velikom greškom. Naime, te akorde je u većini situacija prilično teško razlikovati iz razloga jer se oni vrlo često pojavljuju s potpuno jednakim rezonirajućim tonovima (za sve detalje pozivamo čitatelja da prouči [27]).

3. *Greška 3. vrste: Pogreška paralelni dur i mol*

   Greška 3. vrste pojavljuje se kad je rezultat klasifikacije paralelni durski akord, ako je odgovarajuća oznaka molski akord, odnosno paralelni molski akord ako je odgovarajuća oznaka durski akord. Tu grešku odvajamo od ostalih iz razloga što se paralelne durske i molske ljestvice sastoje od potpuno istih tonova. Iako je njihova uloga u ljestvici različita, samim time je klasifikacijski problem znatno teži.

4. *Greška 4. vrste: Pogreška unutar okoline akorda*

   Greška 4. vrste pojavljuje se kad je rezultat klasifikacije neki drugi od akorda koji pripada okolini akorda danog tonaliteta, promatranoj na način kao što je to opisano u Poglavlju 3.3.2 s razlikom da je u ovom slučaju uključen i smanjeni akord građen na sedmom stupnju (oznaka *d*).

5. *Greška 5. vrste: Ostale greške*

   Greška 5. vrste je bilo koja druga greška koja nije ni jednog od ranije opisanih tipova. Takava greška bi u praksi rezultirala disonancijom te su greške tog tipa one greške koje želimo izbjeći.

3.7.3 Usporedba

Rezultate u sve 4 faze uspoređivali smo koristeći dva pristupa: točnost modela i usporedbu distribucije greške.

Za usporedbu distribucije greške koristili smo podjelu skupa pogrešno klasificiranih primjera u 5 kategorija kao što je opisano u Poglavlju 3.7.2. Točnije, promatrali smo možemo li opovrgnuti hipotezu da pristupi u različitim fazama imaju jednaku distribuciju pogrešnih predikcija s obzirom na dane kategoije. Za taj problem koristili smo *Pearsonov* χ^2 *test homogenosti* (vidi [15]) na razini značajnosti od 5%. Kao mjeru slaganja u predikcijama modela dobivenih u različitim fazama koristili smo κ-statistiku (vidi [10]) kako bismo provjerili koliko se modeli dobiveni u različitim fazama slažu.

Poglavlje 4

Rezultati

U ovom poglavlju opisat ćemo rezultate rada i predstaviti jednu mogućnost primjene dobivenog modela za prepoznavanje akorda.

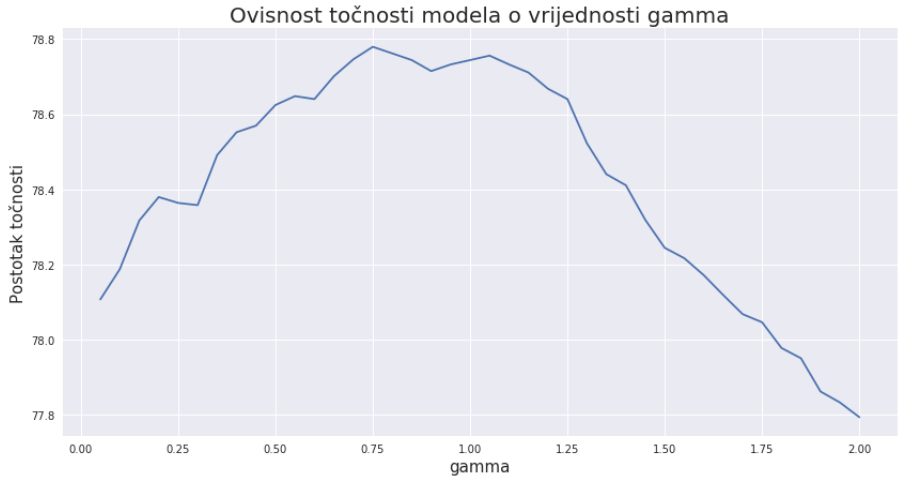
4.1 Evaluacija modela u različitim fazama

U ovom poglavlju će za svaku od četiri faze predstavljene u Poglavlju 3 biti opisani rezultati koji su dobiveni koristeći pristup opisan u toj fazi u kombinaciji sa *SVM* klasifikacijskim algoritmom (vidi Poglavlje 3.6) te će svaka od faza biti uspoređena s prethodnim fazama na način kao što je to opisano u Poglavlju 3.7.3.

4.1.1 Faza 1

U ovoj fazi, skup podataka se sastoji od originalnih 16 atributa koji su opisani u Poglavlju 2. Ideja ova faze bila je izgraditi model koristeći originalni skup podataka kako bismo mogli procijeniti eventualna poboljšanja modela nakon dodavanja novih atributa.

Budući da je u ovoj fazi skup atributa vrlo mali u odnosu na veličinu skupa podataka (51012 primjera na skupu za treniranje), linearni modeli već i na samom skupu za treniranje daju veliku grešku. Stoga se veća kompleksnost modela postiže postavljanjem parametara C i γ na način kako je to opisano u Poglavlju 3.6.2. Točnije, pristranost (*bias*) modela je smanjena podešavanjem parametra C na veliku vrijednost, u našem slučaju, $C = 100000$. Paramater γ birali smo na način da smo za taj fiksan C 10-erostrukom unakrsnom validacijom na skupu za treniranje odredili točnost za vrijednosti γ iz nekog intervala te smo za konačnu vrijednost parametra odabrali onaj za koji je izračunata točnost najveća (vidi Slika 4.1). Izmjerena točnost najveća



Slika 4.1: Točnost dobivena unakrsnom validacijom na skupu za treniranje u prvoj fazi s fiksnim parametrom $C = 100000$ u odnosu na vrijednosti γ

je za $\gamma = 0.07$ i iznosi 78.7462%.

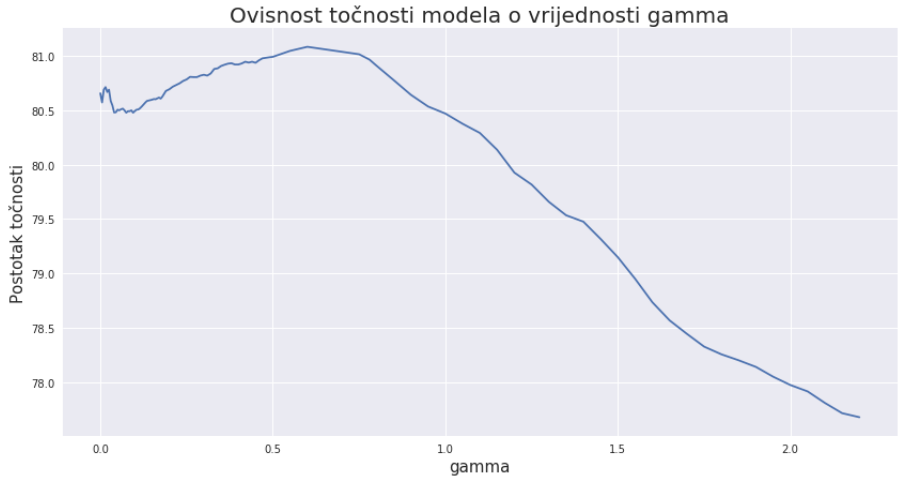
Nakon treniranja modela s navedenim parametrima, model je evaluiran na skupu primjera za testiranje gdje dobivena točnost iznosi 76.9349%. Dobivena točnost nije zadovoljavajuća s obzirom na referentne rezultate opisane u Poglavlju 1.3. Iz tog razloga kompleksnost modela povećavamo dodavanjem novih atributa, što je opisano u kasnijim fazama.

Ovaj model na skupu primjera za testiranje daje 301 krivo predviđeni akord; od toga je 81 njih prve, 4 druge, 30 treće, 95 četvrte i 91 greške pete vrste. Dakle gotovo 27% krivo predviđenih akorda spada u zanemarivu pogrešku prve vrste, što je aspekt značajno poboljšava konačni rezultat ovog modela u smislu povećanja točnosti. Ipak, veliki broj, 30%, pogrešaka je pete vrste, odnosno, 30% pogrešno klasificiranih primjera je potpuno pogrešno klasificirano što je postotak koji u kasnijim fazama želimo smanjiti.

4.1.2 Faza 2

U ovoj fazi dodana su 3 atributa (vidi Poglavlje 3.2) s ciljem dobivanja dodatne informacije o kontekstu kako bi se povećala točnost prilikom klasifikacije primjera. Osnovna ideja ove faze je bila vidjeti koliko atributi generirani na ovaj način, bez korištenja atributa klase u generiranju atributa, poboljšavaju model.

U odnosu na prvu fazu, u drugoj fazu su parametri RGB funkcije postavljeni na drugačiju vrijednost. Budući da želimo povećati kompleksnost modela, fiksirali smo C na manjoj vrijednosti. Empirijski je odabrana vrijednost



Slika 4.2: Točnost dobivena unakrsnom validacijom na skupu za treniranje u drugoj fazi s fiksnim parametrom $C = 1000$ u odnosu na vrijednosti γ

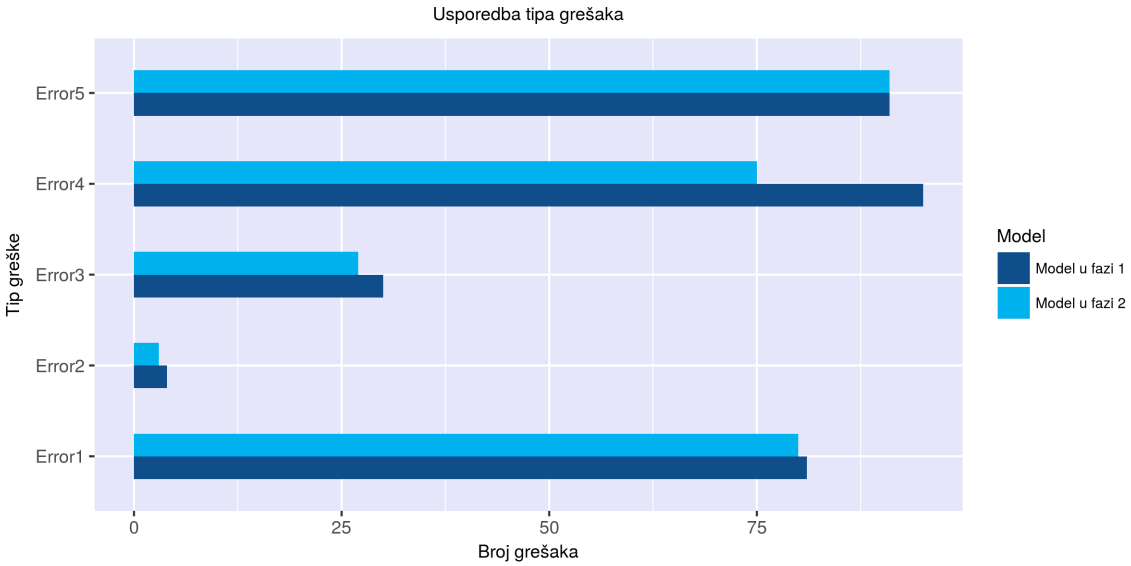
$C = 1000$. Parametar γ smo, kao i u prethodnoj fazi, odabrali 10-erostrukom unakrsnom validacijom na skupu za treniranje. Dobivene točnosti za isprobane vrijednosti γ vidimo na grafu na Slici 4.2. Najbolja točnost postignuta je za $\gamma = 0.60$ i iznosi 81.0829%.

Uočimo da smo već u ovoj fazi uspjeli na 10-erostrukoj unakrsnoj validaciji postići ciljnu točnost. Zaključujemo da su rezultati dobiveni u ovom radu usporedivi s rezultatima postignutim u originalnom radu (vidi [33]). Međutim, potpuna evaluacija predstavljenog modela u odnosu na model predstavljen u [33] zahtjeva detaljniju statističku analizu te potpuniji uvid u metode autora referentnog rada (vidi [33]).

Još jednom naglasimo da je smanjivanje vrijednosti parametara u odnosu na prvu fazu u skladu sa značenjem tih parametara opisanih u Poglavlju 3.6.2. Dodavanjem novih atributa, u odnosu na prvi model, povećala se i kompleksnost tog modela.

Dobivena točnost na skupu podataka za testiranje je 78.8506%. Nadalje, ovaj model na skupu za testiranje pogrešno klasificira 276 primjera. Broj pogrešno predviđenih akorda prema vrsti pogreške se također promijenio: 80 pogrešaka je prve vrste, 3 druge, 27 treće, 75 četvrte i 91 pete vrste. Uočimo da broj pogrešaka prve vrste, koje smatramo zanemarivima, čini gotovo 29% svih pogrešaka što je rezultat koji smatramo izrazito dobrim. No, udio pogrešaka 5. vrste je i dalje velik, čini 33% svih pogrešaka.

Usporedimo sada pristupe u prve dvije faze na način kako je to opisano u Poglavlju 3.7.3. Uočimo da se točnost na skupu za testiranje povećala za točno 25 primjeraka što je približno povećanje od 2% u odnosu na prvu



Slika 4.3: Usporedba tipa greške u fazama 1 i 2

fazu. Iako model u drugoj fazi ima veću točnost u odnosu na model prve faze, dobivena vrijednost κ-statistike za dva promatrana modela iznosi 0.884 što je vrijednost relativno blizu 1. Zaključujemo da se modeli dobro slažu u predikcijama.

Distribucija greške ovisno o vrsti može se vidjeti na grafu na Slici 4.3. Iz samog grafa možemo zaključiti da je broj pogrešaka prve, druge, treće i pete vrste gotovo jednak u obje faze. No, primjećujemo da se broj grešaka četvrte vrste u drugoj fazi značajnije smanjio. Iz samog grafa možemo zaključiti da model nakon druge faze postaje robusniji na grešku četvrte vrste koja se odnosi na akorde koji pripadaju istoj okolini akorda. Ipak, dobivena p-vrijednost za χ^2 test iznosi 0.8124 zbog čega ne možemo odbaciti hipotezu o jednakoj distribuciji grešaka modela u prve dvije faze.

4.1.3 Faza 3

U trećoj fazi, u odnosu na drugu fazu dodana su još tri atributa. Dakle u odnosu na skup podataka korišten u prvoj fazi dodano je ukupno šest atributa. Naglasimo još jedno da, u ovom slučaju, tri novogenerirana atributa ovise u oznaci akorda onih događaja koji se pojavljuju u istoj pjesmi i predhode promatranom događaju (vidi Poglavlje 3.3). Time se simulira slijedni pristup učenju modela. Osnovna ideja proučavanja ove faze je vidjeti poboljšavaju li takvi atributi dosadašnji model.

Budući da je, u odnosu na prvu fazu, dodano šest novih atributa, mo-

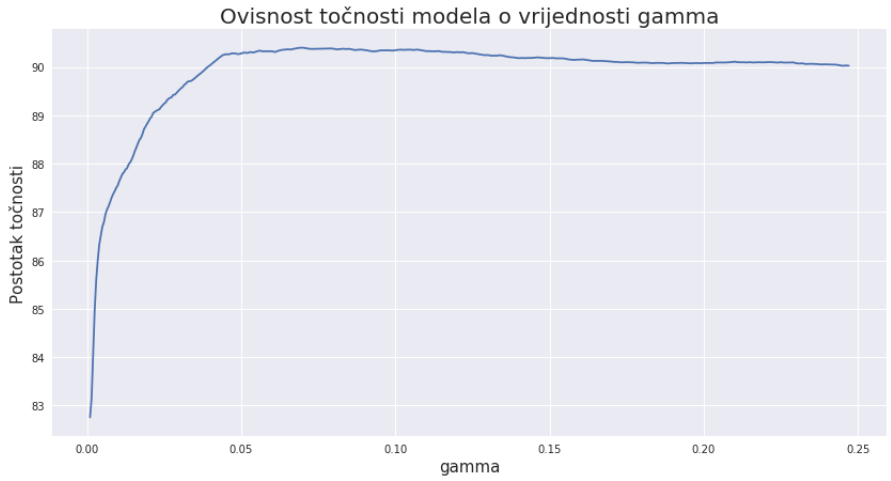
del istreniran parametrima kao u prvoj fazi davao je veliku grešku na skupu za testiranje iako je postizao odlične rezultate na skupu za treniranje. S obzirom na to, grešku dobivenu na skupu za testiranje pripisujemo pretreniranju. Zbog toga je parametar C u ovoj fazi bitno smanjen te je odabranja vrijednost $C = 10$. Parametar γ je, kao i u prethodnim fazama, biran 10-erostrukom unakrsnom validacijom. Najveća točnost dobivena je za $\gamma = 0.07$ te iznosi 90.3964% (vidi graf na Slici 4.4). Dobivena točnost na 10-erostrukoj unakrsnoj validaciji sada se već značajno povećala u odnosu na ciljnu točnosti. Međutim, moramo naglasiti da su u skupu za treniranje sve vrijednosti dodatnih atributa u potpunosti točne dok se u stvarnim primjenama greška u predikciji u nekom događaju propagira i na događaje koje u istoj pjesmi dolaze nakon njega. Bez obzira na to, na skupu za testiranje smo dobili točnost koja iznosi 83.6015% što je najbolje dobivena točnost od svih faza. Poboljšanje u odnosu na početnu fazu iznosi oko 6%. Dobiveno poboljšanje smatramo značajnim jer su, zbog same prirode problema, veće vrijednosti teško dohvatiljve. Zbog svega navedenog, model generiran u ovoj fazi koristit ćemo u kasnijim primjenama.

U ovom slučaju, model je pogrešno klasificirao 214 primjera. Od pogrešno klasificiranih primjera, njih 56 je greška prve vrste, 3 greška druge vrste, 16, treće, 64 četvrte te je 75 primjera potpuno pogrešno klasificirano. Kad se pogleda da je ponovno 26% pogrešnih primjera zanemarivo, rezultat dobiven ovim modelom postaje još značajniji. No pogreška pete vrste je i dalje ostala pogreška najvećeg udjela koji iznosi 35%.

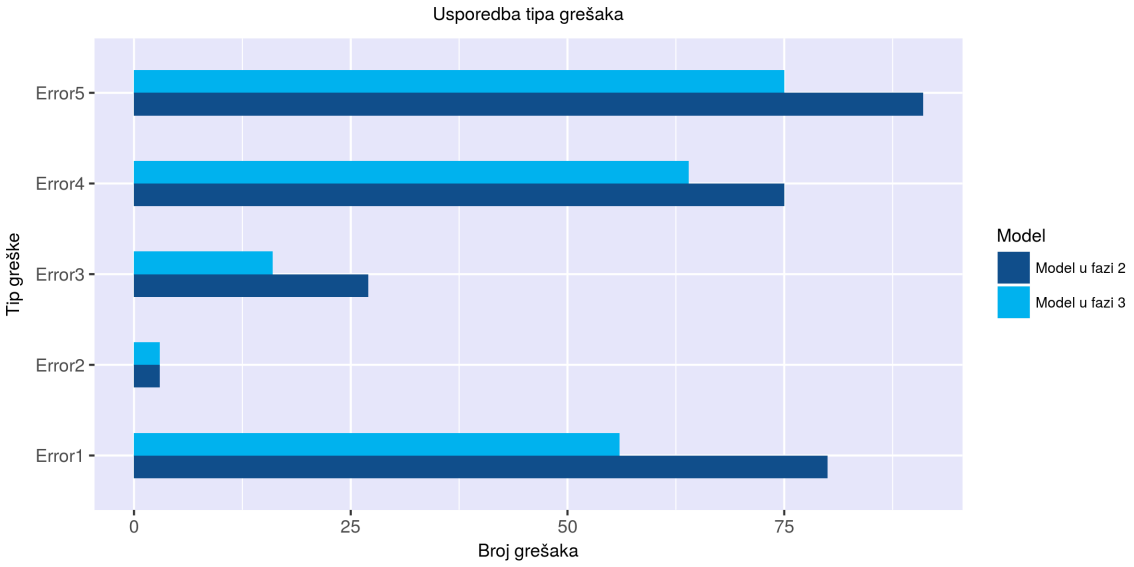
Vrijednost κ-statistike za usporedbu modela druge i treće faze iznosi 0.84 pa ponovno zaključujemo da se modeli u ove dvije faze dobro slažu u predikcijama iako model u trećoj fazi ima veću točnost. Kako se vidi iz samog grafa na Slici 4.5 distribucija greške s obzirom na vrstu greške ne bi se smijela značajnije razlikovati. To nam potvrđuje i χ^2 test čija p-vrijednost u ovom slučaju iznosi 0.7524 pa ne odbacujemo hipotezu da su greške jednako distribuirane. Međutim, uočimo da se u odnosu na prvu fazu p-vrijednost ipak smanjila.

4.1.4 Faza 4

U ovoj fazi, kao što je i navedeno u Poglavlju 3.5 koristimo prilagođeni *word2vec* model kojim nastojimo čim bolje opisati kontekst u kojem se akordi iz skupa za treniranje pojavljuju. Atributi *prethodni akord* i *najčešći akord* više nisu kategorički, nego su predstavljeni 100-dimenzionalnim realnim vektorom.



Slika 4.4: Točnost dobivena unakrsnom validacijom na skupu za treniranje u trećoj fazi s fiksnim parametrom $C = 10$ u odnosu na vrijednosti γ



Slika 4.5: Usporedba tipa greške u fazama 2 i 3

Vizualizacija

U Poglavlju 3.5 već smo naveli da smo dobivene vektorske reprezentacije vizualizirali koristeći *t-SNE* (vidi [21]). Vektore smo preslikali u dvodimenzialni prostor i prikazali na način kako je to prikazano na Slici 4.6. Svih 108 akorda koji se pojavljuju u skupu za treniranje grupirani su u 4 klase M (*dur*), m (*mol*), d (*smanjeni*), *M7* i *m7*. To su ujedno i najfrekventnije oznake akorda. Sve ostale oznake su ubačene u te osnovne klase na način da su svi akordi koji su karakteristični za durski tonalitet (*M4*, *M6* i svi ostali koji sadrže M) svrstani u klasu M. Analogno je učinjeno i za molski tonalitet.

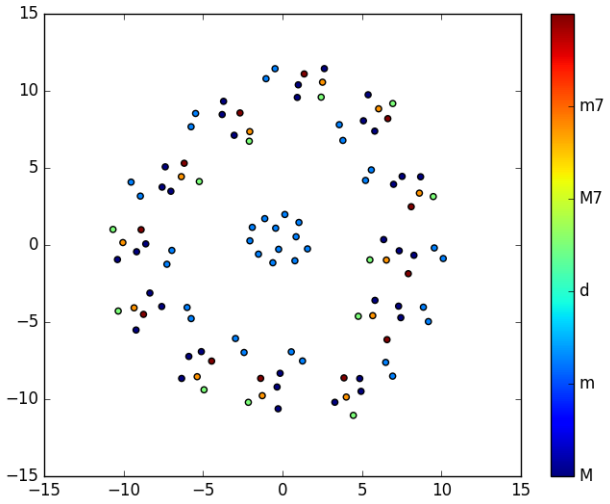
Primjetimo da se lako uočava kružni raspored akorda, kao i konstantne udaljenosti između različitih klasa akorda. Dobivena struktura je izrazito zanimljiva jer nam sugerira da se svi akordi koji pripadaju određenoj klasi preslikavaju na sličan način (norma im je konstantna) samo su, za neki određeni kut, rotirani u odnosu na ishodište. Taj dio je lako opravdati činjenicom da su akordi koji pripadaju istoj klasi jednako građeni (koriste se isti intervali između tonova), jedino sadrže drukčije tonove. Uočimo da je smjerova u koje s vektori protežu upravo 12, onoliko koliko ima različitih tonova. Također, smatramo da utjecaj na dobivenu strukturu ima i proširivanje skupa za treniranje na način koji je opisan u Poglavlju 2.4 gdje smo sve pjesme iz skupa za treniranje proširili na svih 12 mogućih tonova čime smo kontekst od svake pjesme prenijeli na sve tonove. S obzirom na sve navedeno, možemo zaključiti da smo postigli reprezentaciju akorda koja sadrži više informacija o kontekstu akorda od prijašnje kategoričke reprezentacije.

Rezultati

U ovoj fazi se značajnije promijenio skup podataka na kojem smo učili model jer su dva nominalna atributa iz prethodne faze, zamijenjena numeričkim 100-dimenzionalnim vektorom. Ponovno smo, kao i u svim fazama, za klsifikaciju koristili *SVM* algoritam. Za ovu fazu je empirijski odabrana vrijednost $C = 50$. Parametar γ je odabran 10-erostrukom unakrsno validacijom te je najbolja točnost postignuta za $\gamma = 0.006$ te iznosi 88.2949% (vidi graf na Slici 4.7). Mala vrijednost parametra γ je razumljiva budući da je dodan veliki broj novih atributa pa je sammim time dimenzinalnost problema već na početku značajno povećana u odnosu na prethodne faze.

Na skupu za testiranje, postignuta je točnost od 82.2989% koja je ipak manja od postignute točnosti u prethodnoj fazi, no razlika nije statistički značajna. Bliskost dva modela u predikcijama potvrđuje i vrijednost κ statistike koja iznosi 0.898.

Ako promotrimo distribuciju greške prema vrsti, u ovoj fazi broj grešaka



Slika 4.6: Vizualizacija *word2vec* modela pomoću *t-SNE*

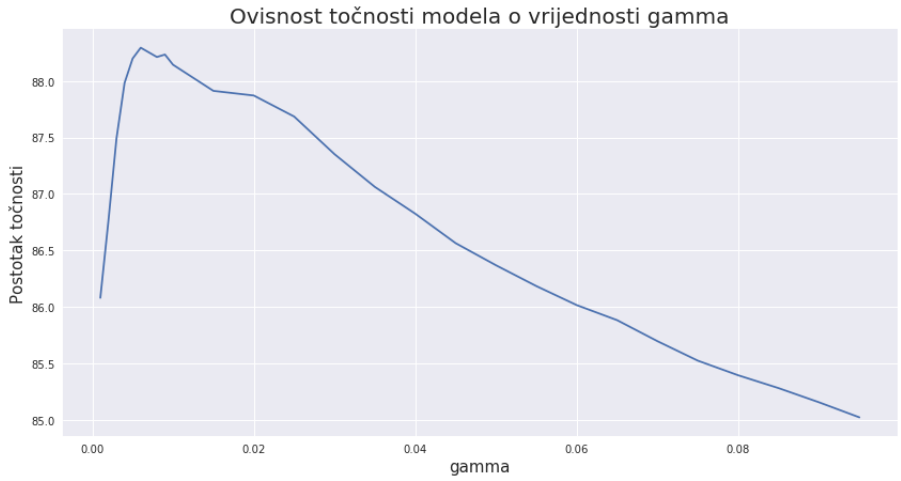
prve vrste je 52, broj grešaka druge vrste 3, broj grešaka treće vrste 20, četvrte 67 i pete 89. Primijećujemo da se broj grešaka pete vrste značajnije povećao u odnosu na prethodne faze te u ovoj fazi on iznosi gotovo 39%. No, kada promatramo p-vrijednost χ^2-testa, koja iznosi 0.8767, ponovno ne možemo odbaciti hipotezu da je distribucija grešaka u obje faze jednaka. Distribuciju greške vidimo na grafu na Slici 4.8.

Spomenimo da je za korištenje algoritma *word2vec* ključna velika količina podataka u skupu za treniranje što u ovom radu nije slučaj. Naime, naš skup za treniranje sadrži samo 44 pjesme (uzmimo u obzir da *word2vec* uči "rečenice", tj. pjesme u našem slučaju). Budući da se s ovako malim skupom za treniranje točnost nije značajnije pokvarila u odnosu na prijašnje faze, smatramo da bi se generiranjem dodatnih primjera za treniranje ovim pristupom mogli postići i bolji rezultati od navedenih. Iz tog razloga smatramo ovaj pristup izrazito zanimljivim i uspješnim te bismo ga svakako uključili u sva daljnja istraživanja vezanih uz ovu temu.

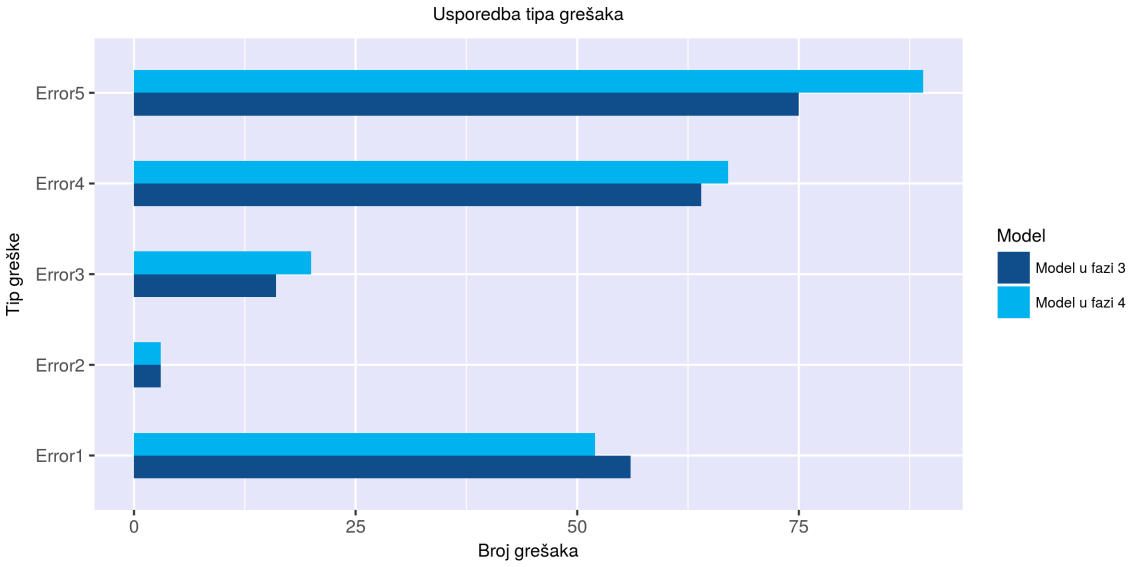
4.2 Primjena modela - programsko rješenje

U nastavku predstavljamo programsko rješenje koje nudi mogućnost prepoznavanja glazbenih akorda koristeći model opisan u ovom radu.

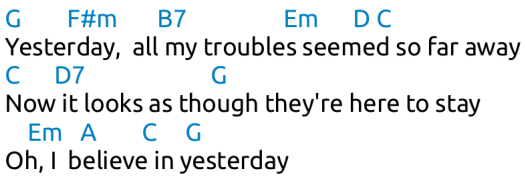
Kako je već navedeno u uvodnim poglavljima, jedna od mogućih primjena modela za prepoznavanje glazbenih akorda je aplikacija koja za dani



Slika 4.7: Točnost dobivena unakrsnom validacijom na skupu za treniranje u četvrtoj fazi s fiksnim parametrom $C = 50$ u odnosu na vrijednosti γ



Slika 4.8: Usporedba tipa greške u fazama 3 i 4



Slika 4.9: Neformalan zapis prve strofe pjesme *Yesterday* grupe *The Beatles*

ulaz, pjesmu zapisanu u nekom formatu, za izlaz daje oznaku akorda koji se pojavljuju u toj pjesmi za neke, točno određene, vremenske intervale. Aplikacije tog tipa mogu biti izrazito korisne u primjeni. Na primjer, služile bi glazbenicima kao odlična referenca prilikom izvođenja pojedine pjesme te bi uštedile sate utrošene na ručno labeliranje odgovarajućih promjena akorda u pjesmi. Također, takve aplikacije bi bile izuzetno dobra podloga za vježbu prilikom učenja sviranja glazbenih instrumenata.

Nadalje, spomenimo da se u glazbi (naročito popularnoj ili *rock* glazbi) često koristi još jedan neformalan zapis koji olakšava izvođenje neke pjesme. Zapis se sastoji od teksta pjesme koja se izvodi nadopunjenog oznakama akorda koji trebaju biti reproducirani u trenutku pjevanja teksta. Oznake akorda su obično napisane iznad riječi na kojoj se promjena dogodila. Takav zapis s formalnog gledišta ima mnogo nedostataka. Na primjer, on ne daje nikakvu informaciju o ritmu, mjeri i ostalim bitnim elementina glazbenog djela te je vezan isključivo uz pjesme koje imaju tekst. Bez obzira na to, takav format je izrazito popularan u primjeni te je mogućnost generiranja takvog zapisa svakako jedna od motivacija za proučavanje ovog problema samim time što se značajno smanjuje mogućnost ljudske pogreške. Na Slici 4.9 vidimo primjer jednog takvog neformalnog zapisa prve strofe pjesme *Yesterday* grupe *The Beatles*.

Aplikacija koja je nastala kao jedan od rezultata ovog rada implementira još jednu moguću primjenu opisanog modela. Osnovna funkcionalnost izrađenog programskog rješenja je mogućnost odabira pjesme u *MIDI* formatu. Odabrana pjesma se pretprocesira tako da se podijeli u događaje (*evente*) na način kako je to opisano u Poglavlju 2. Nakon tog, pretprocesirana pjesma (lista događaja) daje kao ulaz već istreniranom modelu za prepoznavanje akorda koji svakom tom događaju pridružuje odgovarajuću oznaku akorda. Zatim, korisnik može pokrenuti reprodukciju pjesme u integriranom *MIDI* playeru te se istovremenom na grafičkom sučelju prikazuju odgovarajuće oznake akorda u skladu s dijelom pjesme koji se trenutno reproducira. Takav koncept zamišljen je da olakša glazbeniku izvođenje odabrane pjesme na način da računalo automatski određuje promjene akorda te da korisnik zna u

kojem trenutku treba akord biti promijenjen.

U nastavku će biti predstavljeni detalji vezani uz samu implementaciju programskog rješenja.

4.2.1 MIDI format i pretprocesiranje

Programsko rješenje ima mogućnost obrade pjesama koje su isključivo u *MIDI* formatu. *MIDI* je skraćenica za *Musical Instrument Digital Interface*. To je tehnički standard koji opisuje protokol koji se sastoji od *MIDI* poruka različitih tipova koji specificiraju odsviran ton, tempo, glasnoću, vremenski žig te još mnogo različitih stvari nužnih za opis glazbenog djela na najjednostavnijoj tehničkoj razini, poruke koje računalo može procesirati. Detaljno pojašnjavanje *MIDI* formata izlazi iz okvira ovog rada te za sve detalje čitatelja upućujemo na [2].

Taj format je odabran iz razloga što je originalni skup podataka (vidi Poglavlje 2) nastao iz korala Johanna Sebastiana Bacha spremljenih upravo u tom formatu. Također, *MIDI* format je relativno jednostavan za procesiranje i razumijevanje. Procesiranje nekog drugog formata (npr. *MP3*) bi prebacilo fokus rada s promatranog problema, prepoznavanja akorda, na procesiranje ulaznog signala što je izrazito kompleksan problem sam za sebe (vidi [32]). No, to je cilj nekog budućeg proučavanja budući da je ideja da model bude što robusniji na bilo koju vrstu ulaznog signala.

Svaka *MIDI* datoteka se unutar aplikacije pretprocesira na način da se za odabrani vremenski interval skupljaju informacije o svim tonovima koji se pojavljuju u tom vremenskom intervalu, sprema se najdublja nota u tom intervalu (bas nota) te se određuje naglašenost tog vremenskog intervala. Na taj način se specificira jedan događaj kako je to prikazano u Poglavlju 2 koji se kasnije prosljeđuje modelu. Kao vremenski interval trajanja svakog od događaja korisnik aplikacije može odabrati tri opcije: pola dobe[1], jedna doba ili dvije dobe. Duljinu jedne dobe u pjesmi moguće je odrediti procesiranjem odgovarajućih *MIDI* poruka (*set tempo* meta poruka) i informacija u zaglavlju *MIDI* datoteke (vidi [2]). Tonovi koji rezoniraju u svakom od tih intervala dohvaćani su pomoću *note on* i *note off MIDI* poruka koje označavaju početak, odnosno kraj rezoniranja pojedinog tona. Važno je napomenuti da se *MIDI* poruke šalju asinkrono na nekoliko *MIDI kanala* što dodatno otežava parsiranje same *MIDI* datoteke.

U nastavku iznosimo nekoliko tehničkih detalja vezanih uz samu implementaciju. Napomenimo da smo za procesiranje *MIDI* datoteka koristili *MIDO library* za programski jezik *Python*. Pjesme koje aplikacija može procesirati moraju zadovoljavati *General MIDI Specification* (vidi [1]). Također, za reproduciranje *MIDI* datoteka, korisnik mora na računalu imati instaliran

i pokrenut *TiMidity++ software synthesizer* (vidi [4]).

4.2.2 Implementacija modela

Modeli razvijani u svim fazama rada (vidi Poglavlje 3) objedinjeni su u *Java* aplikaciju pod nazivom *ChordRecognition* te su implementirani koristeći isključivo programski jezik *Java* (verzija 1.8). Arhitektura samog rješenja u potpunosti slijedi opis svake pojedine faze te je prikazana na Slici 4.10 na kojoj vidimo reducirani *UML class dijagram*. Kao što je vidljivo iz slike, arhitektura aplikacije je relativno komplicirana te sadrži mnogo implementacijskih detalja koje nećemo posebno pojašnjavati jer nisu toliko vezani uz sam problem koji opisujemo u ovom radu. Napomenimo da smo za treniranje modela *SVM* algoritmom koristili *LIBSVM* implementaciju algoritma (vidi [8]) unutar *Weka frameworka* (vidi [5]). Svi modeli koji se mogu koristiti za klasifikaciju su već istrenirani sa zadanim parametrima na način kako je opisano ranije u ovom poglavlju te spremljeni u odgovarajuće datoteke koje aplikacija po potrebi učitava u memoriju kao *Classifier* objekte.

Kao što smo već naveli u prethodnim poglavljima, kao model za prepoznavanje akorda u programskom rješenju odabrali smo model koji je rezultat treće faze u ovom radu (vidi Poglavlje 3.3 i Poglavlje 4.1.3).

4.2.3 Opis rada programskog rješenja

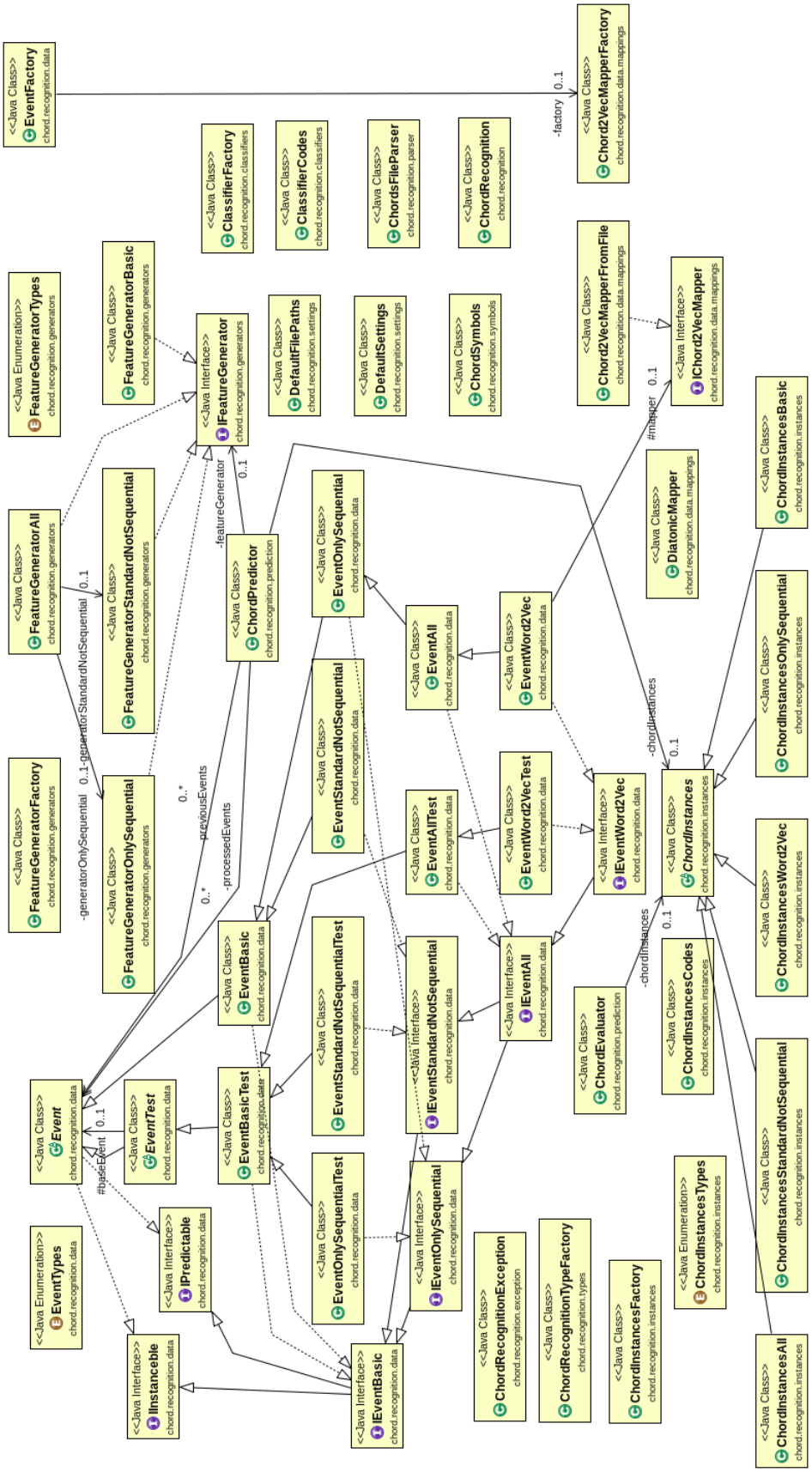
U ovom poglavlju predstavit ćemo korištenje i funkcionalnosti implementiranog programskog rješenja kroz jednu korisničku sesiju. Korisničko sučelje implementirano je u programskom jeziku *Python* koristeći *PyQt5* (vidi [3]).

Sesija će biti prikazana kroz nekoliko koraka.

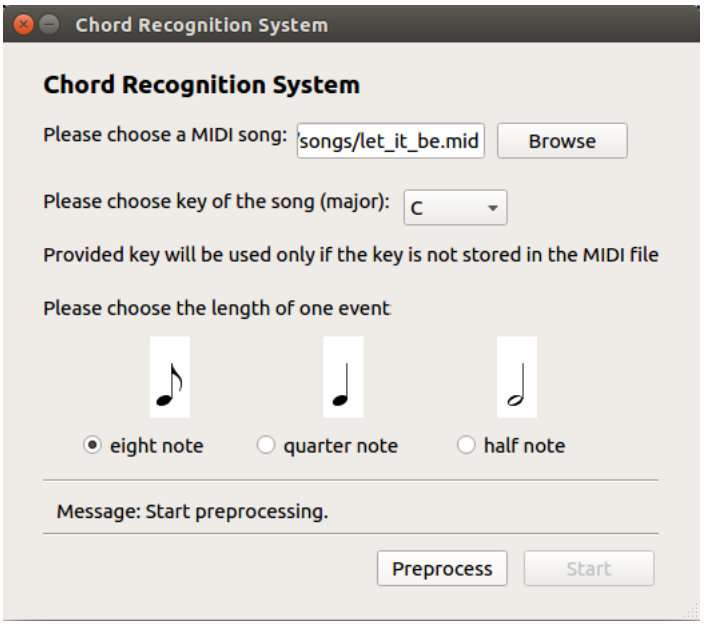
1. Odabir pjesme u *MIDI* formatu

   Korisnik pomoću standardnog *Browse* gumba na početnom prozoru aplikacije otvara dijaloški okvir u kojem odabire željenu *MIDI* datoteku. Nakon toga, korisnik može odabrati još dvije opcije. On može odabrati tonalitet odabrane pjesme, tj. može sugerirati modelu koji tonalitet da koristi. Važno je napomenuti da ako u odabranoj *MIDI* datoteci postoji *key signature* poruka koristit će se tonalitet već zapisan u *MIDI* datoteci, dakle, korisnikov izbor će biti zanemaren. Nadalje, korisnik može odabrati koliko će trajati jedan događaj (*event*), tj. može odrediti koliko će trajati vremenski interval na razini kojeg će se određivati odgovarajuća oznaka akorda. Moguće opcije su, kao što je već

[1] Doba označava osnovnu jedinici vremena u pjesmi, često se neformalno definira kao ritam koji glazbenici "otkucavaju" prilikom slušanja ili izvođenja pjesme.



Slika 4.10: Reducirani *UML class dijagram Java* aplikacije *ChordRecognition*

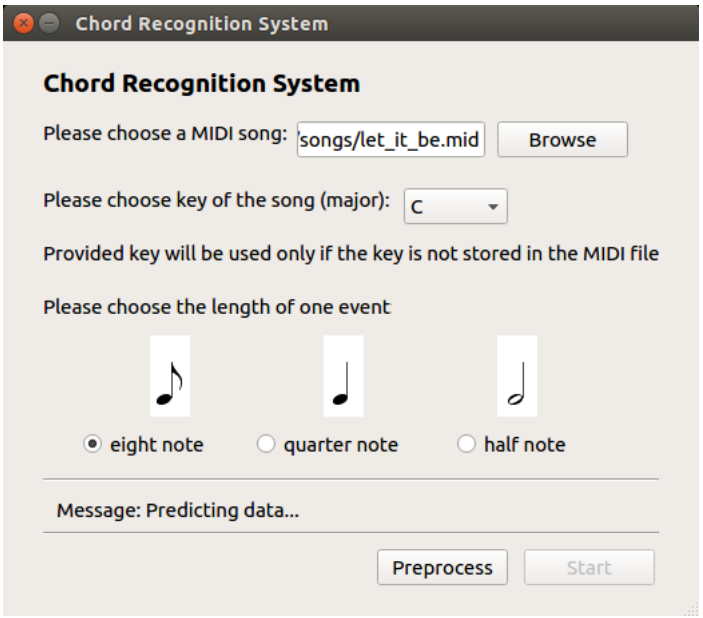


Slika 4.11: Početni prozor programskog rješenja koji koristi model opisan u radu

navedeno u Poglavlju 4.2.1, pola dobe (odabir *osminke*, ♪), jedna doba (odabir *četvrtinke*, ♩) te dvije dobe (odabir *polovinke*, 𝅗𝅥). Na Slici 4.11 vidimo početni prozor aplikacije u kojem se odabiru gore navedene opcije.

2. Pokretanje pretprocesiranja

   Nakon odabira *MIDI* datoteke i podešavanja odgovarajućih parametara korisnik pokreće pretprocesiranje pritiskom na gumb *Preprocess*. Pritiskom na taj gumb odabrana datoteka se pretprocesira, dijeli se na događaje na način kako je to opisano u Poglavlju 4.2.1 te se poziva *Java* aplikacija *ChordRecognition* s odgovarajućim argumentima kojima se generirana lista događaja prosljeđuje modelu za prepoznavanje akorda (vidi 4.2.2). Kao što smo već nekoliko puta naveli, u programskom rješenju se kao model za prepoznavanje akorda koristi model koji je rezultat treće faze opisane u ovom radu (vidi Poglavlje 3.3). Samo predviđanje akorda odvija se na isti način kao što je opisano u Algoritmu 1, dakle simulira se slijedni pristup. Najprije se za svaki događaj generiraju svi potrebni dodatni atributi te se zatim tako definiran događaj prosljeđuje već istreniranom *SVM* klasifikatoru koji određuje odgovarajuću oznaku akorda. Na Slici 4.12 vidimo početni prozor aplikacije tijekom generiranja predikcija. Nakon predikcije, aplikacija interno pridružuje odgovarajuću oznaku akorda svakom događaju te su događaji



Slika 4.12: Početni prozor programskog rješenja tijekom određivanja predikcija

spremni za reproduciranje.

3. Pokretanje *MIDI* playera

   Nakon pretprocesiranja, sve je spremno za reproduciranje *MIDI* datoteke, zajedno s predviđenim oznakama akorda. Pritiskom na gumb *Start* u zasebnom dijaloškom okviru pokreće se *MIDI* player sa standardnim opcijama *play*, *pause* i *stop*. Pritiskom na gumb *play* pokreće se reprodukcija odabrane pjesme s odgovarajućim oznakama akorda prikazanih u prostoru ispod naslova pjesme (vidi Poglavlje 4.13). Još jednom napominjemo da za reproduciranje *MIDI* datoteka treba nezavisno od aplikacije imati pokrenut *TiMidity MIDI Server* (vidi [4]).

4.2.4 Komentar

U ovom opisu izostavljena je većina implementacijskih detalja kao što su definicije leksičke i sintaksne analize *MIDI* datoteke ili pozivanje dijelova aplikacije u zasebnim dretvama da bi se osiguralo asinkrono izvršavanje pojedinih dijelova koda. Napomenimo da je za pokretanje aplikacije nužno instalirati *Python 3.5* te *Qt framework*, točnije, *PyQt5*, *Python* sučelje prema *Qt frameworku* (vidi [3]).

U trenutnoj fazi, aplikacija je još uvijek u *demo* verziji te služi primarno kao demonstracija rezultata ovog istraživanja. Bez obzira na to, ona zasad

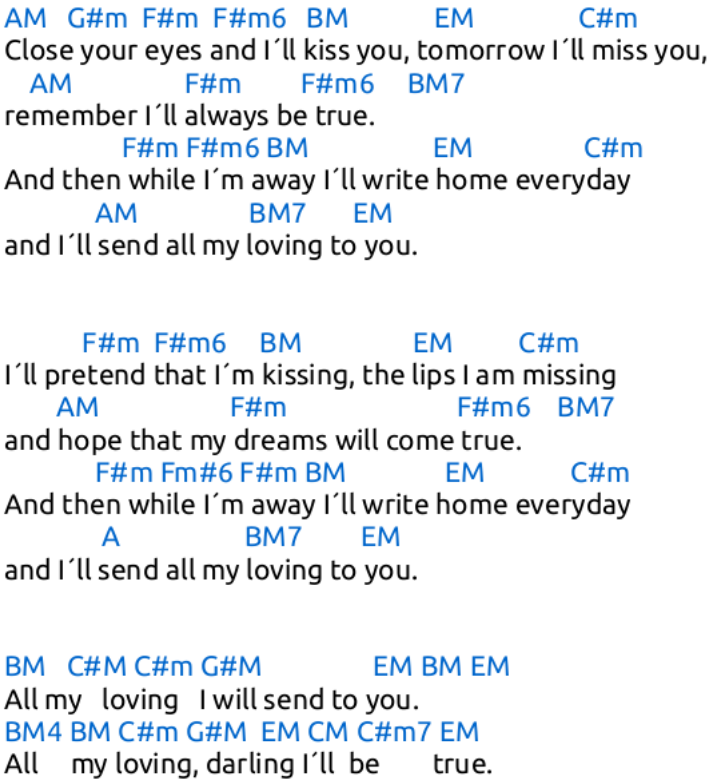
(a) C dur akord

(b) G dur akord

(c) A mol akord

(d) F dur akord

Slika 4.13: Prikaz različitih oznaka akorda tijekom izvođenja pjesme *Let It Be* grupe *The Beatles* u *MIDI* formatu



Slika 4.14: Rezultati dobiveni predstavljenim programskim rješenjem za pjesmu *All My Loving* grupe *The Beatles*

daje izrazito dobre rezultate na pjesmama grupe *The Beatles* koji su usporedivi s rezultatima koje daju slične primjene kao što su web sustav *Chordify*[2]. No, za daljne usporedbe s postojećim modelima i anotacijama potrebna je detaljnija statistička analiza. Kako je naše programsko rješenje u ovom trenutku isključivo ograničeno na *MIDI* datoteke, za usporedbu je najprije potrebno model proširiti na pjesme u prozivoljnom formatu, što izlazi iz okvira ovog rada te će biti predmet nekog budućeg istraživanja.

Primjer rezultata dobivenih našim programskim rješenjem za pjesmu *All My Loving* grupe *The Beatles* ručno prevedenih u gore opisani neformalni zapis vidimo na Slici 4.14.

[2] `https://chordify.net/`

Poglavlje 5

Zaključak i buduća istraživanja

Problem predviđanja akorda kao problem strojnog učenja je specifičan u smislu odabira odgovarajućeg modela i u smislu analize greške iz razloga što isti akordi u različitom kontekstu mogu imati različitu ulogu. S ciljem dobivanja maksimalne moguće točnosti u ovom radu predloženo je nekoliko pristupa koji se primarno sastoje od dodavanja novih atributa s različitim svojstvima te gradnje modela na novodobivenim (proširenim) skupovima atributa koristeći *Support Vector Machines* algoritam (*SVM*). Dakle, pristup u ovom radu usmjeren je na nadograđivanje modela korištenjem novih atributa u svakoj od faza rada, zadržavajući isti klasifikacijski algoritam.

Prijašnji radovi na tu temu koji koriste isti početni skup podataka, skup pjesama J. S. Bacha (vidi Poglavlje 2), autora Radicionija i Esposita (vidi [33]) promatraju problem prepoznavanja akorda kao *Sequential leraning problem* te, u skladu s tim, koriste standardne tehnike sekvencijalnog učenja kao što su *Hidden Markov models* te modifikacije *Viterbijevog algoritma*. Našim pristupom željeli smo pokazati u kojoj mjeri se rezultati mogu poboljšati proširivanjem informacije o kontekstu ugrađivanjem novih atributa u model i koristeći standardni klasifikacijski algoritam, *SVM*. Kao referentnu točnost, zbog istog početnog skupa podataka, uzimali smo upravo rezultate predstavljene u navedenom radu [33].

Nakon pretprocesiranja i dodavanja novih primjera u skup podataka za treniranje (vidi Poglavlje 3), rad smo podijelili u četiri faze. U prvoj fazi, krenuli smo od jednostavnog modela s minimalnim promjenama u odnosu na originalni skup podataka. Već s minimalnim promjenama, koristeći *SVM* algoritam, dobili smo rezultate koji su usporedivi s rezultatima koje su naveli Radicioni i Esposito u svom istraživanju (vidi [33]). U drugoj fazi, s ciljem proširivanja informacije o kontekstu, iz početnih atributa generirali smo tri nova atributa (vidi Poglavlje 3.2). Već u toj fazi rezultat se poboljšao u odnosu na prvu fazu te je na 10-erostrukoj unakrsnoj validaciji postignuta

veća točnost od referentne točnosti. Međutim, potpuna evaluacija modela predstavljenog u ovom radu u odnosu na model predstavljen u [33] zahtjeva detaljniju statističku analizu te potpuniji uvid u metode autora referentnog rada (vidi [33]).

Treća faza (vidi Poglavlje 3.3) uključivala je temeljitije istraživanje prirode problema i uvođenje atributa koji će profiniti znanje i omogućiti modelu robusnost na rubne slučajeve. U toj fazi simuliramo sekvencijalni pristup učenju na način da za gradnju novih atributa za svaki događaj koristimo već klasificirane događaje iz iste pjesme. Takav pristup je prirodan u smislu mogućih primjena generiranog modela jer je ideja da se svi događaji u nekoj pjesmi procesiraju sekvencijalno. Na taj način smo u sam proces učenja uključili i informaciju o nekim karakteristikama same pjesme, nismo jedan događaj promatrali kao zasebni entitet. Ovo je i faza u kojoj smo ujedno postigli i najveću točnost na skupu za testiranje, koja iznosi 83.6015% te smo iz tog razloga model generiran u ovaj fazi koristili u kasnijim primjenama (vidi 4.2).

U četvrtoj fazi, isprobali smo primjenu postojećih rješenja koja se koriste u drugim domenama strojnog učenja koji za osnovni cilj imaju prepoznavanje konteksta (vidi 3.5). Preciznije koristili smo tehniku iz problema procesiranja prirodnog jezika koja se naziva *word2vec* (vidi [29]). Iako rezultati te faze nemaju najveću točnost, ova faza predstavlja veliki zaokret u dosadašnjim pristupima budući da omogućava da se kontekst akorda iz svih pojavljivanja u skupu za testiranje ugradi u sam atribut u obliku k-dimenzionalnog vektora. No, glavna pretpostavka za dobre rezultate *word2vec* pristupa je da se taj model uči na velikom broju primjera (u slučaju procesiranja jezika, govorimo o nekoliko desetaka tisuća primjera). Iako u našem slučaju koristimo svega četrdesetak primjera pjesama, *word2vec* pristupom uspjeli smo se približiti rezultatima postignutima u trećoj fazi ovog rada. Iz tog razloga smatramo da bi taj pristup mogao donijeti značajna poboljšanja u nekim budućim istraživanjima promatranog problema.

U Poglavlju 4.2 predstavili smo jedno moguće programsko rješenje koje koristi model generiran u trećoj fazi ovog rada. Aplikacija na ulazu prima pjesmu u *MIDI* formatu, dijeli je na događaje (odgovarajuće vremenske intervale) te koristeći generirani model, svakom događaju pridružuje odgovarajuću oznaku akorda. Nakon toga, programsko rješenje omogućuje reproduciranje odabrane pjesme uz informaciju o promjeni akorda u trenutku u kojem se dogodila. Jedan od ciljeva budućih istraživanja je predloženi model proširiti na pjesme zapisane u proizvoljnom formatu, bez ograničavanja na *MIDI* format.

Zahvale

Zahvaljujemo se dr. sc. Tomislavu Šmucu na stručnoj i velikodušnoj pomoći prilikom ovog istraživanja. Veliko hvala i doc. dr. sc. Goranki Nogo na odličnim savjetima i komentarima te velikoj potpori. Posebno se zahvaljujemo Mateju Mihelčiću i Matiji Piškorcu što su svojim stručnim savjetima i mišljenjima neprekidno bili uz naš rad od samog početka.

Bibliografija

[1] *General MIDI specification*, `https://www.midi.org/specifications/category/gm-specifications`, posjećena 19.4.2017.

[2] *MIDI association homepage*, `https://www.midi.org/`, posjećena 18.4.2017.

[3] *PyQt*, `https://riverbankcomputing.com/software/pyqt/intro`, posjećena 19.4.2017.

[4] *TiMidity*, `https://wiki.archlinux.org/index.php/timidity#Installation`, posjećena 15.3.2017.

[5] Machine Learning Group at the University of Waikato, *Weka 3: Data Mining Software in Java*, `http://www.cs.waikato.ac.nz/ml/weka/`, posjećena 19.4.2017.

[6] Bruce Benward, *Music in Theory and Practice Volume 1*, McGraw-Hill Higher Education, 2014.

[7] Bernhard E. Boser, Isabelle M. Guyon i Vladimir N. Vapnik, *A training algorithm for optimal margin classifiers*, Proceedings of the fifth annual workshop on Computational learning theory, ACM, 1992, str. 144–152.

[8] Chih Chung Chang i Chih Jen Lin, *LIBSVM: A library for support vector machines*, ACM Transactions on Intelligent Systems and Technology **2** (2011), 27:1–27:27, Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[9] Heng-Tze Cheng, Yi-Hsuan Yang, Yu-Ching Lin, I-Bin Liao i Homer H. Chen, *Automatic chord recognition for music classification and retrieval*, Multimedia and Expo, 2008 IEEE International Conference on, IEEE, 2008, str. 1505–1508.

[10] Jacob Cohen, *A coefficient of agreement for nominal scales*, Educational and psychological measurement **20** (1960), br. 1, 37–46.

[11] Daniele P. Radicioni i Roberto Esposito, *Bach Chorales Harmony Data Set*, `https://archive.ics.uci.edu/ml/datasets/Bach+Choral+Harmony`, posjećena 7.4.2016.

[12] Roberto Esposito i Daniele P. Radicioni, *Carpediem: Optimizing the viterbi algorithm and applications to supervised sequential learning*, Journal of Machine Learning Research **10** (2009), br. Aug., 1851–1880.

[13] Jerome Friedman, Trevor Hastie i Robert Tibshirani, *The elements of statistical learning*, sv. 1, Springer series in statistics Springer, Berlin, 2001.

[14] Takuya Fujishima, *Realtime Chord Recognition of Musical Sound: a System Using Common Lisp Music.*, ICMC, 1999, str. 464–467.

[15] Priscilla E. Greenwood i Michael S. Nikulin, *A guide to chi-squared testing*, sv. 280, John Wiley & Sons, 1996.

[16] Christopher Harte i Mark Sandler, *Automatic chord recognition using quantised chroma and harmonic change segmentation*, Centre for Digital Music, Queen Mary University of London (2009).

[17] S. Sathiya Keerthi i Chih Jen Lin, *Asymptotic behaviors of support vector machines with Gaussian kernel*, Neural computation **15** (2003), br. 7, 1667–1689.

[18] Richard Lawn i Jeffrey L. Hellmer, *Jazz: theory and practice*, Wadsworth Pub Co, 1993.

[19] Kyogu Lee, *Automatic Chord Recognition from Audio Using Enhanced Pitch Class Profile.*, ICMC, 2006.

[20] Kyogu Lee i Malcolm Slaney, *Automatic Chord Recognition from Audio Using a HMM with Supervised Learning.*, ISMIR, 2006, str. 133–137.

[21] Laurens van der Maaten i Geoffrey Hinton, *Visualizing data using t-SNE*, Journal of Machine Learning Research **9** (2008), br. Nov., 2579–2605.

[22] Sephora Madjiheurem, Lizhen Qu i Christian Walder, *Chord2Vec: Learning Musical Chord Embeddings*, (2016), `http://www.cs.nott.ac.uk/~psztg/cml/2016/papers/CML2016_paper_5.pdf`.

[23] Robert L. Marshall i Walter Emery, *Johann Sebastian Bach*, `https://www.britannica.com/biography/Johann-Sebastian-Bach`, posjećena 20.4.2017.

[24] Matthias Mauch, *Automatic chord transcription from audio using computational models of musical context*, (2010).

[25] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado i Jeff Dean, *Distributed representations of words and phrases and their compositionality*, Advances in neural information processing systems, 2013, str. 3111–3119.

[26] Barrie Nettles, *Harmony I*, Berklee College of Music, 1987.

[27] Barrie Nettles i Richard Graf, *The chord scale theory & jazz harmony*, Advance music, 1997.

[28] The Editors of Encyclopædia Britannica, *Chorale*, `https://www.britannica.com/art/chorale`, posjećena 20.4.2017.

[29] Christopher Olah, *Deep Learning, NLP, and Representations*, `http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/`, posjećena 11.4.2017.

[30] Walter Piston, *Harmony. Rev. and expanded by Mark DeVoto*, New York: WW Norton, 1978.

[31] William H. Press, Saul A. Teukolsky, William T. Vetterling i Brian P. Flannery, *Numerical Recipes: The Art of Scientific Computing (3rd edition)*, New York: Cambridge University Press, 2007.

[32] Lawrence R. Rabiner i Bernard Gold, *Theory and application of digital signal processing*, Englewood Cliffs, NJ, Prentice-Hall, Inc., 1975. 777 p. **1** (1975).

[33] Daniele P. Radicioni i Roberto Esposito, *BREVE: An HMPerceptron-Based Chord Recognition System*, Advances in Music Information Retrieval, Springer, 2010, str. 143–164.

[34] Don Michael Randel, *The Harvard dictionary of music*, sv. 16, Harvard University Press, 2003.

[35] Alexander Sheh i Daniel P. W. Ellis, *Chord segmentation and recognition using EM-trained hidden Markov models*, (2003).

[36] Deeplearning4j Development Team, *Deeplearning4j: Open-source distributed deep learning for the JVM, Apache Software Foundation License 2.0*, `https://deeplearning4j.org/word2vec`, posjećena 11.4.2017.

[37] David A. Van Dyk i Xiao Li Meng, *The art of data augmentation*, Journal of Computational and Graphical Statistics **10** (2001), br. 1, 1–50.

[38] Vladimir N. Vapnik, *The nature of statistical learning theory*, Springer science & business media, 2013.

Sažetak

Ana Paliska i Filip Srnec
Prepoznavanje glazbenih akorda koristeći tehnike strojnog učenja

Glazba je oduvijek važan dio ljudskog života te, kao takva, sadrži izazove čijim bi se rješavanjem olakšali mnogi glazbeni zadaci. Jedan od takvih izazova je prepoznavanje glazenih akorda proizvoljne pjesme koji je u ovom radu prikazan kao klasifikacijski problem strojnog učenja.

Ono što čini ovaj problem kompleksnim je činjenica da različiti akordi u različitom kontekstu mogu imati različitu uloge. Iz tog razloga, ključ rješavanja problema je na odgovarajući način u model uključiti informaciju o kontekstu u kojem se neki akord pojavljuje. Većina prijašnjih radova na ovu temu, koriste tehnike sekvencijalnog učenja. U ovom radu, ideja je koristiti drugačiji pristup te informaciju o kontekstu u model ugraditi generiranjem novih atributa. Rad je podijeljen u četiri faze. Prva faza sastoji od klasifikacije na originalnom skupu podataka, druga faza proširuje početni skup atributa dodatnim atributima dok treća faza simulira sekvencijalni pristup korištenjem prethodnih događaja. U četvrtoj fazi, za proširenje informacije o kontektsu koristimo distribuiranu reprezentaciju zasnovanu na *word2vec* algoritmu. Za klasifikaciju koristimo algoritam *Support Vector Machines (SVM)*.

Također, predloženo je programsko rješenje koje na ulazu prima pjesmu u *MIDI* formatu, dijeli ju na događaje, te koristeći model generiran u trećoj fazi ovog rada pridružuje tim događajima odgovarajuće oznake akorda.

Ključne riječi: Prepoznavanje akorda, Strojno učenje, Klasifikacija, SVM, Word2Vec.

Summary

Ana Paliska i Filip Srnec
Music Chord Recognition Using Machine Learning Techniques

Music has always been involved in everyday life. As such, there are some unsolved challenges whose solution could be used to facilitate many problems related to music. One of those challenges is chord recognition which is presented in this paper as a machine learning classification problem.

The fact that different chords may have different meaning in different context, is the main complexity of this problem. This is why a good solution to this problem should include appropriate context information about chord appearance. Most of related works for solving this problem use sequential learning techniques and algorithms. Instead of using sequential learning algorithms, we decided to use a bit different approach to context modelling and preserve context information by generating new features. This paper is divided in four phases. The first phase includes classification on a slightly modified original dataset. The second phase extends the dataset from the first phase with additional features. The third phase simulates sequential approach by using previous events. Last, the fourth phase uses dense representation based on the *word2vec* algorithm to expand context information. In all phases the *Support Vector Machines* algorithm is used for classification.

Besides problem modelling, this paper also includes a program solution. It expects a song in *MIDI* format as input, parses the song to a sequence of events and uses the model described in third phase to predict appropriate chord labels.

Key words: Chord Recognition, Machine Learning, Classification, SVM, Word2Vec.