—

Prepoznavanje spam SMS poruka

Ognjen Stipetić
Grgur Valentić
Vedrana Vazdar

Sažetak

U ovom radu izradili smo klasifikator za odvajanje neželjenih (*spam*) od pravih (*ham*) poruka koji bi mogao biti korišten za automatsko cenzuriranje spama. Koristili smo UCI Machine Learning Repository bazu spam poruka na kojoj je već napravljeno nekoliko sličnih radova, što nam daje priliku za usporedbu razultata. Osim riječi koje se pojavljuju u poruci, kao feature smo koristili i dužinu poruke, nepismenost, udio brojeva u poruci te udio nealfanumeričkih znakova. Isprobali smo algoritme klasifikacije koji su preporučeni u literaturi – Naive Bayes, K-nn, SVM i neuronske mreže. Neuronske mreže su, usprkos najadužem vremenu izvršavanja, dale najlošije rezultate, pa smo zaključili da nisu primjenjive na ovaj problem. Algoritmi SVM i 1-nn su puno točniji i imaju jako nizak udio zaustavljenog hama. Naive Bayes je također prilično točan te ima nešto viši udio zaustavljenog hama, ali i puno viši udio ulovljenog spama. Najbolje rezultate dobili smo koristeći SVM s Gaussovim kernelom, koji je na testnom setu zaustavio 0.33% hama i ulovio 89.2% spama.

Uvod

Problem neželjenih SMS poruka u zadnje se vrijeme pojavljuje sve češće. Razlog tome je ponajviše njihova sve manja cijena, ali i (zbog neočekivanosti spama) velik broj odgovora korisnika. U Kini je, primjerice, prošle godine udio neželjenih SMS poruka iznosio čak 20 - 30%, što pokazuje kako je ovo vrlo aktualan i važan problem. [4]

Glavni cilj ovog projekta je razviti klasifikator koji će uspješno raspoznavati neželjene, odnosno spam SMS poruke od ostatka "pravih" poruka. Uvriježena terminologija za "prave" poruke za koje se pretpostavlja da ih korisnici žele pročitati je "ham" SMS, te ćemo i mi uvažiti tu terminologiju. Popravili smo neke od rezultata iz literature[1][2][3]. Budući da je problem SMS spama znatno manji od problema e-mail spama, očekivano, postoji manji broj radova na tu temu. Iako detekcija e-mail i SMS spama ima dosta dodirnih točaka, glavna zamjerka metodama klasifikacije za e-mail spam, kao u [3] je ta što ne uzimaju kao značajku duljinu poruke. Naime, za očekivati je da će pošiljatelji spam SMS-a htjeti iskoristiti što je više moguće znakova budući da plaćaju jednaku naknadu dokle god je duljina poruke ispod 160 znakova. Također, osim uvriježenih značajki (pojavljivanje ključnih riječi) koristili smo i neke koje se u literaturi nisu pojavljivale, kao recimo pismenost poruke (pretpostavili smo da će se pošiljatelji spam poruka paziti da ostavljaju razmak iza točke i zareza, dok će prave poruke često biti krivo napisane) te udio nealfanumeričkih znakova. Problem koji smo uočili kod postojećih radova je taj što kao pokazatelj uspješnosti klasifikatora ističe točnost (obično oko 90%), što zaista nije čudno, obzirom da vjerujemo da je udio ham poruka u svim porukama iznad 90%, barem za poruke u Velikoj Britaniji odakle je naš skup podataka. Stoga ćemo, osim na točnost, paziti i na udio zaustavljenog hama i udio zaustavljenog spama, posebnu pozornost pridajući udjelu zaustavljenog hama (jer je najbitnije da korisnici prime sve prave poruke, pa makar morali ponekad primiti i poneki spam).

Prikupljanje podataka

Kao bazu podataka uzeli smo kolekciju SMS poruka od UCI Machine Learning Repositoryja [5]. Baza se sastoji od 5574 SMS poruka, od čega 4827 ham te 747 spam poruka na engleskom jeziku, a isječak iz skupa svih poruka možete vidjeti u tablici:

ham	As I entered my cabin my PA said, " Happy B'day Boss !!". I felt special. She askd me 4 lunch. After lunch she invited me to her apartment. We went there.
spam	You are a winner U have been specially selected 2 receive £1000 or a 4* holiday (flights inc) speak to a live operator 2 claim 0871277810910p/min (18+)
ham	Goodo! Yes we must speak friday - egg-potato ratio for tortilla needed!
ham	Hmm...my uncle just informed me that he's paying the school directly. So pls buy food.
spam	PRIVATE! Your 2004 Account Statement for 07742676969 shows 786 unredeemed Bonus Points. To claim call 08719180248 Identifier Code: 45239 Expires

Iz ovih poruka smo prvo izvukli nekoliko atributa koji se baziraju na broju i vrsti znakova u porukama, a ne uključuju pojavljivanje ključne riječi a to su:

- broj riječi (prirodan broj) - riječi smo odvajali po svim nealfabetskim znakovima, pa tako npr. „We'll" postaju dvije riječi. Očekujemo da će spam poruke uobičajeno imati veći broj riječi obzirom da žele iskoristiti maksimalni kapacitet poruke koji je ograničen na 160 znakova.
- duljina (prirodan broj) - ukupan broj znakova. Smatramo da će ovo biti dobar indikator iz istog razloga kao i broj riječi.
- nepismenost (1 ili 0) - oznaka 1 ako u poruci postoji alfabetski znak iza točke ili zareza, inače 0. Ovo bi mogao biti dobar indikator za ham jer je za očekivati da će pošiljatelji spam poruka paziti da poruka bude informatički pismena.
- Udio brojeva u poruci (realni broj između 0 i 1) – očekujemo da će ova mjera indicirati spam jer će se u spam porukama često pojavljivati telefonski brojevi i novčani iznosi

- udio nealfanumeričkih znakova (realan broj između 0 i 1) - slično kao i za prethodni atribut, smatramo da će ovo biti dobar indikator za spam.

Druga važna klasa atributa koju smo izvukli iz poruka su ključne riječi. Prvo smo izostavili iz poruka sve nealfabetske znakove te razdvojili riječi na svakom pojavljivanju nealfabetskog znaka, i sve znakove prebacili u lower case.

Zatim smo napravili svojevrstan „word stemming" - proces poistovjećivanja riječi istoga korijena. Ideja je da želimo postovjetiti riječi poput „teacher" i „teaching", jer ako je jedna od njih dobar indikator za spam ili ham, za očekivati je da će biti i druga. Iako ovaj algoritam već postoji implemetiran, u sklopu NLTK bibilioteke za Python [6], odlučili smo napraviti slabiju verziju ovog algoritma budući da smo cijeli ostatak preprocesiranja podataka napravili u C-u, a nismo našli već razvijen pogodan alat za C. Stoga smo odlučili napraviti jednostavniju verziju algoritma koja naprosto uzima prva četiri znaka riječi, te ako je riječ kraća, dopuni ju do četiri znaka znakom '_'. To dopunjavanje je napravljeno zbog kasnijeg lakšeg baratanja. Iako zvuči pomalo naivno, algoritam je našao ključne riječi koje su bile zadovoljavajuće. Ipak ovo smatramo jednom komponentom projekta koja se da unaprijediti. Nakon stemminga, ogledna tablica s početka poprima ovaj oblik (0 za ham, 1 za spam):

0 as__ i___ ente my__ cabi my__ pa__ said happ b___ day_ boss i___ felt spec she_ askd me__ lunc afte lunc she_ invi me__ to__ her_ apar we__ went ther
1 you_ are_ a___ winn u___ have been spec sele rece or__ a___ holi flig inc_ spea to__ a___ live oper clai p___ min_
0 good yes_ we__ must spea frid egg_ pota rati for_ tort need
0 hmm_ my__ uncl just info me__ that he__ s___ payi the_ scho dire so__ pls_ buy_ food
1 priv your acco stat for_ show unre bonu poin to__ clai call iden code expi

Iz ovakvog teksta odabrane su ključne riječi po sljedećem pravilu. Za svaku smo riječ izračunali:

$$f_{spam} = \frac{\#pojavljivanja\ u\ spamu}{\#spam\ poruka}$$

$$f_{ham} = \frac{\#pojavljivanja\ u\ hamu}{\#ham\ poruka}$$

Te, ukoliko je za danu riječ $f_{spam}/f_{ham} \geq 2$ ili $f_{ham}/f_{spam} \geq 2$, označi riječ kao ključnu (u slučaju da je uvjet prvog omjera ispunjen riječ ćemo nazivati „spam kandidatom", a u slučaju drugog „ham kandidatom"), te će dani omjer (onaj koji je veći od 2) odgovarati nekoj mjeri „koliko je riječ ključna", te taj broj nazovimo sa f_{key}. Primjetimo da ova mjera zaista ima smisla, jer kada u izrazima za f_{spam} i f_{ham} ne bismo dijelili s brojem spam (ham) poruka, dogodilo bi se da bi puno riječi bilo dobar indikator za ham naprosto zato jer ima puno više ham poruka u bazi.

Napomenimo još kako smo izbacili riječi koje se pojavljuju previše rijetko. Npr. riječ koja se jednom pojavi u hamu i niti jednom u spamu bi imala $f_{key} = f_{ham}/f_{spam} = \infty$, što očito nije zadovoljavajuće, stoga smo postavili f_key = 0, za sve riječi za koje je broj pojavljivanja te riječi u kategoriji koju ona indicira (spam ili ham) manji od 0.1% ukupnog broja riječi u toj kategoriji. U našem slučaju ovo je značilo da u obzir ulaze riječi koje imaju barem 14 pojavljivanja u spamovima odnosno barem 57 u hamovima. Napomenimo ovdje usput da, iako je omjer broja spam i ham poruka ukupno 747/4827 = 0.15, ovaj omjer granica od 14/57 = 0.25, nije zabrinjavajuć, te štoviše, ide u prilog hipotezi kako su spam poruke u pravilu duže. Usput rečeno, 38 riječi je prošlo granicu broja pojavljivanja, a imalo je mjeru $f_{key} = \infty$, odnosno uopće se nije pojavljivalo u jednoj od kategorija. Poredak ključnih riječi prema mjeri će nam kasnije zaista biti bitan, budući da ćemo kao parametre modela uzimati broj atributa, no nikad nećemo uzeti manje od 38 ključnih riječi za atribute, pa nam poredak među tih „najvažnijih" 38 neće biti bitan.

Granice za omjer $f_{key} \geq 2$, te 0.1% ukupnog broja riječi po kategoriji odabrane su isprobavanjem kako bismo dobili zadovoljavajuće velik broj ključnih riječi, ali isto tako kako bismo dobili riječi koje nam se čine koliko toliko smislenima.

Traženje ključnih riječi napravili smo na prvih 4500 poruka (nakon što smo slučajno ispermutirali sve poruke), koje ćemo kasnije koristit za trening i kros-validaciju. Preostalih 1174 poruka koristili smo za test, te ih nismo koristili za traženje ključnih riječi. Primjetimo da će ovo rezultirati malim overfittingom na kros-validacijskom skupu, budući da su ključne riječi izvađene jednom, i to iz svih 4500 poruka. Ipak, smatramo da će ovaj overfitting biti dovoljno mali budući da očekujemo vrlo slične ključne riječi generirane iz svih 4500 poruka, kao i one koje bi bile generirane iz svakog trening skupa posebno. Dodatno, kada bismo iz svakog trening skupa našli posebne ključne riječi, onda bismo morali na neki način odlučiti koje će nam biti ključne riječi koje ćemo koristiti za testnu skupinu što bi dodatno zakompliciralo stvari. A budući da ovime nismo napravili nikakav overfitting za testnu skupinu, odlučili smo se za ovaj pristup.

Među 4500 poruka koje smo koristili bilo je 3910 hamova i 590 spamova, te je ovaj algoritam našao 143 spam kandidata i 95 ham kandidata. Iako kasnije neće biti bitno je li riječ kandidat za spam ili ham, već samo njena f_{key} mjera, zanimljivo je vidjeti kako ima uvjerljivo više spam kandidata nego ham, iako bi algoritam sam po sebi trebao izbaciti podjedanko kandidata obje kategorije. Ipak, ovako nešto ima smisla ako uzmemo u obzir samu prirodu spam i ham poruka. Za očekivati da će u spam porukama biti dosta riječi koje se koriste izrazito često, poput „prize", „winner", i slično, dok će u ham porukama riječi biti ravnomjernije raspoređene.

Prije nego iznesemo rezultate ovog algoritma, napomenimo da smo u literaturi [2] naišli na ideju korištenja bigrama, što bi se dalo generalizirati na n-grame [4] kao

feature. Ukratko, algoritam traži koji parovi ili n-torke riječi koje se zajedno pojavljuju češće nego što bi bilo očekivano. I ovaj algoritam implementiran je u spomenutoj biblioteci NLTK [6], no zbog jednostavnosti smo upotrijebili algoritam koji smo opisali, a budući da smo dobili rezultate koji se poklapaju s intuicijom, nismo se upuštali u implementaciju tih složenijih algoritama, iako ovo vidimo kao još jedno potencijalno poboljšanje. Evo rezultata:

redni broj	riječ	pojavljivanja u hamu	pojavljivanja u spamu	f_{key}
1.	tone	0	73	∞
2.	he__	181	0	∞
3.	priz	0	73	∞
4.	she_	143	0	∞
5.	guar	0	43	∞
6.	lor_	126	0	∞
...	...	...	...	...
38.	gud_	60	0	∞
39.	clai	1	88	583.186
40.	uk__	1	57	377.746
...	...	...	...	...
237.	them	67	5	2.022
238.	im__	67	5	2.022

Sada vidimo da su među top 40 spam kandidata riječi „tone", „prize", „guaranteed", „claim", „uk". Sve ovdje nabrojane riječi su očekivane. Riječ „uk" se pojavila od internet adresa, a preostale tri navedene su klasične spam riječi. Detaljnijom analizom vidjeli smo i da su i riječi „nokia" i „ringtone" spam kandidati. Ovo upućuje na problem koji nije u našem dosegu rješavanja, čini se da je baza nereprezentativan uzorak poruka, jer iako je baza poruka napravljena 2012. [5], jasno da Nokia nije jedini proizvođač mobitela.

Slično možemo primjetiti i za ham poruke - neki od top kandidata su „he", „she", „lor", „gud", „them", „im". Riječi „he", „she" i „them" su očekivane jer su to uobičajene riječi, „gud" i „im" ide u prilog argumentu da ham poruke češće koriste nepismene izraze - analizirajući same poruke vidimo da ove riječi zapravo znače „good" odnosno „I'm".

Uočili smo da su jako dobri indikatori za ham riječi „lt", „gt" i „amp", štoviše, te se riječi vrlo često pojavljuju u ham porukama, a niti jednom u spamu. Pročitavši nekoliko poruka koje ih sadrže, zaključili smo da su iz nekog razloga brojevi u ham porukama često zamijenjeni s „`&lt;#&gt;`", primjerice u poruci „`I'm going for bath will msg you next &lt;#&gt; min`", dok se u svim spam porukama brojevi ispravno prikazuju. Nakon što smo probali otvoriti file s porukama u 5 različitih programa na 3 računala i 2 različita operativna sustava, zaključili smo da je problem zbilja u samoj bazi podataka. Budući da je to vrlo vjerojatno bug samo u ovoj bazi podataka, odlučili smo da naši algoritmi neće koristiti te feature u učenju jer ih u praksi ne bismo imali. Zato smo zamijenili te ključne riječi običnim brojevima kako ne bismo „varali", ali da udio brojeva i nealfanumeričkih znakova ostane isti kakav je bio u samim porukama. Moguće je da postoje još neke klase pogrešaka koje nismo otkrili (jer i ove smo otkrili čistom igrom slučaja) koje još više doprinesu boljem rezultatu algoritama na testnom setu, ili ga pak možda pogoršaju, iako smatramo da je poboljšanje vjerojatnije. Stoga, ocjena ovih algoritama će biti optimistična, te bi na uređenijoj bazi algoritmi koji su ovdje korišteni vrlo vjerojatno pokazivali nešto lošije rezultate. Ipak to ne znači da će rezultati dobiveni na ovoj bazi biti potpuno nerelevantni, te vjerujemo da bi isti algoritmi trenirani na boljim podacima dali samo donekle lošije rezultate.

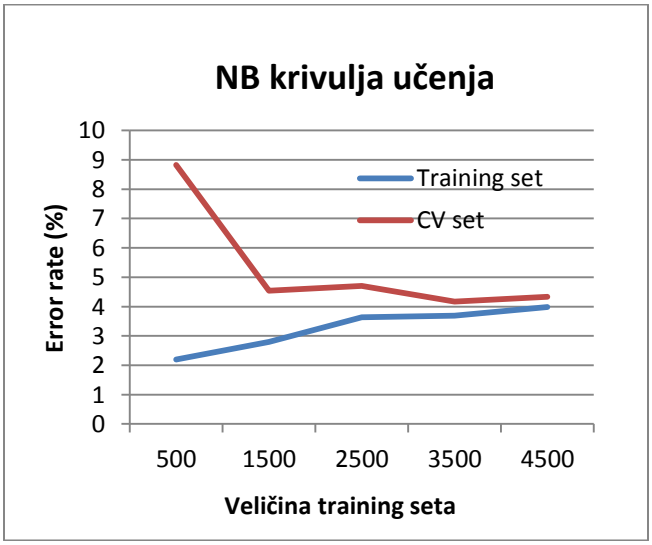
Iako su radovi iz literature [1][2][3] rađeni na istoj bazi podataka, oni nigdje ne spominju te probleme, tako da je moguće da su koristili te informacije koje ne bi imali u praksi, to jest njihovi algoritmi možda bolje rade na ovoj bazi podataka nego što bi radili na novim, ispravno formatiranim, podacima.

Algoritmi strojnog učenja

U nastavku opisujemo rezultate dobivene raznim metodama strojnog učenja. Koristili smo implementacije svih algoritama iz RapidMinera 6.4. Isprobali smo algoritme Naive Bayes, SVM (s linearnim i Gaussovim kernelom), K-nn (ispostavilo se da najbolje radi za K=1 i K=3) te neuronske mreže.

Naive Bayes

Prvi algoritam koji smo isprobali je onaj koji se u literaturi[1] spominje kao najbolji, dakle Naive Bayes. Kako bismo provjerili je li nam training set dovoljno velik, odnosno ima li naš algoritam probema s biasom/underfittingom odnosno varijancom/overfittingom, napravili smo learning curve.



Na x-osi se nalazi veličina dijela training seta koji smo koristili, dok se na y-osi nalazi učestalost pogreške,

odnosno 100% – accuracy. Crvena linija označava kros-validacijsku pogrešku, dok plava crta označava pogrešku na istom setu na kojem je treniran. Budući da se crvena i plava linija približavaju na desnom dijelu grafa, te vidimo da nije došlo do overfittinga.

Kada testiramo na test setu, dobijemo sljedeću matricu konfuzije:

	true 0	true 1	class precision
pred. 0	875	10	98.87%
pred. 1	42	147	77.78%
class recall	95.42%	93.63%	
accuracy	95.16%		

Ovi su rezultati nešto lošiji od Naive Bayes klasifikatora u [1] , ali znatno bolji nego u [2] i [3] . Ne znamo je li to posljedica različitih featurea koje smo odabrali, lošije implementacije algoritma ili promijenjenog dataseta (to jest, nedostatka „<#>“ featurea). Također, vidimo da ovaj klasifikator ima nedopustivo visok postotak zaustavljenog hama – za gotovo 5% ham poruka kaže da su spam, što znači da potencijalni korisnici ne bi primili svaku dvadesetu poruku pravu poruku, što ga čini neprimjenjivim u praksi.

Support vector machine

Sljedeći algoritam koji smo odlučili isprobati je SVM - Support vector machine. U literaturi ovaj algoritam daje dosta dobre rezultate, iako lošije od Naive Bayesa.

Koristimo LibSVM operator u RapidMineru u kojem treba postaviti nekoliko parametara.

Za dobre rezultate vrlo je važno izabrati dobar kernel i optimalne parametre za podatke koje imamo. S obzirom na odnos veličine training seta i broja feature-a literatura preporučuje radial basis function (gaussian) kernel definiran ovako:

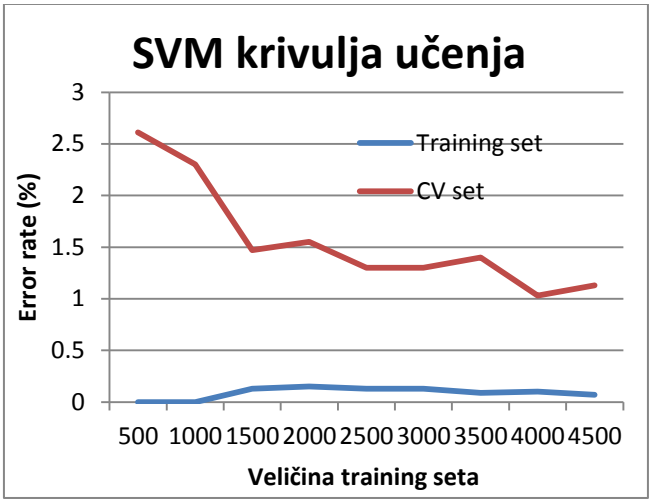
$$K(x_1, x_2) = \exp(-\gamma ||x_1 - x_2||_2^2)$$

Optimalne parametre C i gamma izabrali smo na temelju performansa algoritma dobivenog cross-validacijom za različite kombinacije parametara. Literatura preporuča isprobati kombinacije parametara C i γ na logaritamskoj skali od 0.01 do 1000 za oba parametra. To smo učinili i dobili optimalne parametre γ = 0.03 i C = 15. Zatim smo isprobali algoritam sa još 36 kombinacija u bližoj okolini tih parametara i dobili optimalne parametre γ = 0.07 i C = 15 s kojima smo dalje radili.

Naravno, atributi su normalizirani kako bi svi bili istog reda veličine. U tablici su dani rezultati algoritma kao prosjek 5-fold kros-validacije:

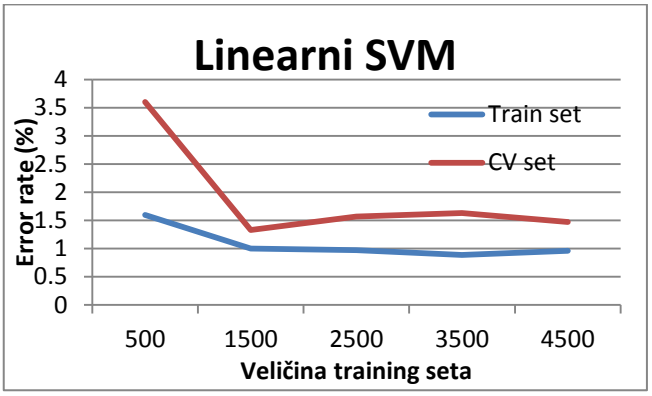
	true 0	true 1	class precision
pred. 0	3900	41	98.96%
pred. 1	10	549	98.21%
class recall	99.74%	93.05%	
accuracy	98.86%		

Kad pogledamo learning krivulju uočavamo da smo malo overfittali podatke. S obzirom da je regularizacijski parametar C odabran kao optimalan na temelju rezultata cross-validacije, njegovo smanjivanje (radi smanjivanja varijance) samo će pogoršati rezultate algoritma.



Druga opcija je da pokušamo smanjiti broj feature-a. Uklonili smo 55 feature-a koji imaju najmanji f_{key} te ponovo pokrenuli algoritam. No, rezultati cross-validacije su sada puno lošiji nego prije (točnost: 91.44%, ulovljeno spama: 35%, zaustavljeno hama: 0.1%) Zaključujemo da uklanjanje featura nema smisla te bi najbolji pristup problemu overfittinga bio prikupiti još podataka, što nažalost ne možemo.

Isprobali smo i linearni kernel s parametrom C=0, dobivenim optimizacijskim postupkom opisanim gore, te dobili sljedeće rezultate i krivulju učenja, također 5-fold kros-validacijom.



	true 0	true 1	class precision
pred. 0	3901	57	98.56%
pred. 1	9	533	98.34%
class recall	99.77%	90.34%	
accuracy	98.53%		

Vidimo da, iako za linearni kernel nije došlo do overfittinga, gaussian kernel daje bolje rezultate i na kros-validacijskom setu, što sugerira da je gaussian zbilja bolji, ali bi mogao postati još bolji kada bi došli do još podataka.

Naš gaussian SVM daje bolje rezultate nego drugi naši algoritmi, ali daje i bolje razultate nego bilo koji algoritam iz literature, tako da možemo reći da je ovaj dio projekta bio prilično uspješan.

Konačno, SVM s gaussian kernelom i parametrima C = 15 i γ = 0.07 tesiramo na test setu.

	true 0	true 1	class precision
pred. 0	914	17	98.17%
pred. 1	3	140	97.90%
class recall	99.67%	89.17%	
accuracy	98.14%		

K-nearest neigbors

Proveli smo i K-nn algoritam koristeći RapidMinerovu funkciju za optimizaciju parametra K i pokazalo se da je najbolja verzija za K=1, a sljedeća najbolja za K=3. Nije iznenađujuće da su mali brojevi najbolji izbor za K jer je set dominiran ham primjerima, pa bi za dovoljno velik K, algoritam jednostavno sve poruke proglasio hamom. Nema smisla crtati krivulju učenja za 1-nn jer bi na training setu 1-nn bio savršen klasifikator – svakom elementu training seta bi najbliži susjed bio on sam, pa bi uvijek točno predvidio koji primjeri su iz kojeg seta. 1-nn nam je dao sljedeće rezultate:

	true 0	true 1	class precision
pred. 0	3894	145	96.41%
pred. 1	16	445	96.53%
class recall	99.59%	75.42%	
accuracy	96.42%		

dok nam je 3-nn dao ovakve rezultate:

	true 0	true 1	class precision
pred. 0	3910	207	94.97%
pred. 1	0	383	100.00%
class recall	100.00%	64.92%	
accuracy	95.4%		

Vidimo da su i 1-nn i 3-nn relativno dobri klasifikatori za ovaj problem, pogotovo je odlično što imaju ekstremno nizak udio zaustavljenog hama – na CV setu 3-nn nije niti jednu ham poruku proglasio spamom. To bi također mogla biti posljedica nejednake zastupljenosti hama i spama u training setu, pa je za svaku novu poruku iz CV odnosno test seta puno lakše naći ham nego spam poruke u blizini. To se slaže s relativno niskim postotkom ulovljenih spamova. Sve u svemu, smatramo da je ovaj klasifikator s gledišta strojnog učenja nešto lošiji od SVM-a, ali bi se u praksi mogao jako dobro primjenjivati jer jako rijetko cenzurira pravu ham poruku.

Na test setu smo dobili sljedeće rezultate (prvo za 1- pa za 3-nn):

	true 0	true 1	class precision
pred. 0	916	35	96.32%
pred. 1	1	122	99.19%
class recall	99.89%	77.71%	
accuracy	96.65	%	

	true 0	true 1	class precision
pred. 0	917	45	95.32%
pred. 1	0	112	100.00%
class recall	100.00%	71.34%	
accuracy	95.81%		

Vidimo da i na ovom setu oba algoritma imaju odličan udio ulovljenog hama i ne toliko dobar udio propuštenog spama.

Zaključak

U tablici prikazujemo točnost, postotak ulovljenog spama i postotak krivo klasificiranog hama za isprobane algoritme koji daju smislene rezultate.

	Točnost	Ulovljeno spama	Zaustavljeno hama
SVM	98.14%	89.17%	0.33%
K-nn	96.96%	77.71%	0.11%
NB	95.16%	93.63%	4.58%

Vidimo da je SVM s gaussian kernelom najbolji klasifikator te iako learning krivulja pokazuje overfitting, na test setu i dalje pokazuje najmanju grešku od svih algoritama. U literaturi [1][2][3] SVM daje rezultate slične ovima.

Naive Bayes ima najveći postotak uhvaćenih spamova, no nedopustivo visok postotak krivo klasificiranih hamova te je stoga u praksi neprimjenjiv. U literaturi NB daje znatno bolje rezultate nego što smo mi dobili, tj. uhvati približno isto spamova, ali krivo klasificira znatno manje ham-ova. Pretpostavljamo da bi dodavanje novih značajki poput često pojavljivanih n-grama znatno poboljšalo uspjeh ovog algoritma te je ovo ideja koju bismo u budućnosti svakako isprobali. Još jedna mogućnost je povećati treshold u NB algoritmu tako da češće predviđa ham-ove nego sad. No, tada će i broj uhvaćenih spamova biti manji čime bi izgubili jedinu prednost ovog algoritma nad drugima.

K-nn, iako krivo klasificira najmanje ham-ova ipak ne uhvati dovoljno spamova da bi ga proglasili dobrim algoritmom, te je SVM u usporedbi s njim značajno bolji. Također, k-nn u literaturi daje vrlo slične rezultate kao naš algoritam.

Pažljivom analizom ključnih riječi koje je naš algoritam odabrao uočili smo, kao što je ranije opisano, da je ova baza vrlo nereprezentativan uzorak općenitih SMS poruka. Trudili smo se ukloniti značajke koje bi mogle davati predobre rezultate, odnosno koje su striktno vezane uz ovu bazu - poput pojavljivanja „<#>“ umjesto brojeva u ham porukama i slično. S obzirom da se drugi radovi nisu toliko bavili analizom značajki smatramo da su njihovi rezultati nerealni i na nekom drugom skupu za testiranje iz druge baze bi davali lošije rezultate. Stoga činjenicu da naši algoritmi daju slične rezultate kao u literaturi smatramo velikim uspjehom.

Literatura

[1] - Houshmand Shirani-Mehr, “SMS Spam Detection using Machine Learning Approach"
[2] - Dr. Ghulam Mujtaba, Majid Yasin, “SMS Spam Detection Using Simple Message Content Features”, Journal of Basic and Applied Scientific Research
[3] - T. Hamsapriya, D. Karthika Renuka, M. Raja Chakkaravarthi, “Spam classification basing on supervised learning using machine learning techniques”, Ictat Journal on Communication Technology
[4] - http://lingpipe-blog.com/2008/05/28/collocations-chi-squared-independence-and-n-gram-count-boundary-conditions/
[5] - http://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection
[6] - http://www.nltk.org/
[7] - http://www.urbandictionary.com/define.php?term=lor