Intelligens spamszűrő

DIPLOMAMUNKA

Készítette: Sass Bálint
ELTE programtervező matematikus szak, nappali tagozat

Témavezető: Porkoláb Zoltán

Budapest, 2003.

„I don't like spam!"
— Monty Python's Flying Circus [26]

Tartalomjegyzék

1 Bevezetés

Egy európai bizottsági jelentés szerint a spam által okozott felesleges kiadás 10 milliárd € (2500 milliárd Ft) évente [9].

Az évek óta folyó spamellenes harc ellenére kijelenthetjük, hogy aki kapott már emailt, az kapott spamet is. Szinte mindenki ismeri a kéretlen leveleket, ha az angol kifejezést kevesen is ismerik: „Ja persze, elolvasás nélkül le szoktam őket törölni."

Az utóbbi tíz évben terjedt el a kéretlen reklámlevelek jelensége az interneten. A problémát többen az internet legégetőbb gondjának tartják [20, 42]. A kéretlen levél kellemetlenség a felhasználónak, és kidobott pénz az internetszolgáltatónak.

Témavezetőm C++ óráján felkeltette a figyelmemet *Markov*-láncokra alapuló példaprogramjával, mely egy kisebb időjárásjelentés-gyűjtemény alapján újabb, értelmesnek tűnő időjárásjelentéseket állított elő. Régóta érdekelte, hogy hogyan lehetne a módszert „megfordítani", azaz a generáló program helyett felismerő programot készíteni. Világos volt, hogy ennek egyik alkalmazási területe éppen a spamszűrés lehet.

2002. augusztusában publikálta az interneten a sokak [16, 35, 47] által meghatározó jelentőségűnek tartott 'A Plan for Spam' című dolgozatát *Paul Graham* [12]. *Graham* szógyakoriságok alapján következtet, és a továbbfejlesztési lehetőségek között megemlíti, hogy egyes szavak helyett szópárokkal is lehetne dolgozni (vö. 6. fejezet).

Ez éppen egybeesett témavezetőm ötletével, így dolgozatom célkitűzése a *Graham*-féle spamszűrő módszer *Markov*-láncos továbbfejlesztése lett.

A markovi továbbfejlesztésről bebizonyosodott, hogy azonos méretű tudásbázis mellett az eredeti szűrőnél gyengébben működik. Ennek más irányú továbbfejlesztésével viszont kellően hatékony szűrőt kaptam, ami bizonyos esetekben kiemelkedő teljesítményt nyújtott (vö. 6.3.3. és 6.4.1. fejezet).

Dolgozatomban bemutatom a spamjelenséget (ld. 2. fejezet), a spamellenes küzdelem módszereit (ld. 3. fejezet). Áttekintem azokat a szövegosztályozási módszereket (ld. 4. fejezet), melyek a másodikgenerációs statisztikai szűrők kifejlesztéséhez vezettek (ld. 5. fejezet). Végül a saját magam által továbbfejlesztett spamszűrőt mutatom be (ld. 6. fejezet).

A dolgozat keretében elkészült program lényegében megfelel a témabejelentőben vállalt követelményeknek: Valószínűségi alapon, az emailek általános tulajdonságai alapján végzi a szűrést. A mesterséges intelligencia eszköztárából egy leginkább a naív bayesi osztályozóhoz hasonlító módszert alkalmaz. Perl nyelven készült, illetve egy része Perl és C++ nyelven párhuzamosan. Tudásbázisa a beérkező levelek alapján továbbfejleszthető, így alkalmazkodni

tud a folyamatosan változó email forgalomhoz.

A program mellett a dolgozat összefoglalja az egész spam-problémakört.

Köszönettel tartozom az ELTE Informatika Tanszékcsoport részéről *Porkoláb Zoltán* témavezetőmnek, aki az eredeti ötletet adta, illetve biztatott, lelkesített a dolgozat elkészítése során; *Vajda Ferenc*nek és *Bedő Sándor*nak, akik rendelkezésemre bocsátották teljes levelezésüket tesztelés céljára; végül szűkebb családom tagjainak, akik a dolgozat gondos lektorálását végezték.

2 A spam

2.1 A spam, mint olyan

2.1.1 Definíció

Egy amerikai bíró mondta egyszer a pornográfiáról [39]: „Definiálni nem tudom, de felismerem, mikor látom". Valahogy így van ez a spammel is, általában a felhasználó ránézésre – többnyire már a feladó és a tárgy alapján – el tudja dönteni, hogy spamet kapott-e, a mindenre kiterjedő definíció megadása azonban nem olyan könnyű.

A spam klasszikus definíciója *kéretlen tömeges email* (UBE: Unsolicited Bulk Email), illetve *kéretlen kereskedelmi email* (UCE: Unsolicited Commercial Email) [42].

Paul Graham nem ért egyet azzal, hogy a spam azonos lenne a kéretlen kereskedelmi emaillel. Nyilvánvaló, hogy ha egy ismeretlentől pont egy olyan terméket kínáló levelet kapok, amire régóta szükségem van, akkor nem bosszankodni fogok, hanem örülni. Lényegesnek tartja, hogy a spameket *automatikusan*, nagy számban állítják elő, ezért a *kéretlen automata email* (UAE: Unsolicited Automated Email) definíciót javasolja [12]. Ez közel áll a fenti UBE definícióhoz.

Fontos, hogy megfogalmazzunk egy olyan átfogó és időtálló definíciót, ami nem hagy kiskapukat a spam bizonyos formáinak, és kellően szabatos ahhoz, hogy a spammel foglalkozó jogi szabályozás alapjául szolgáljon. Ez a meghatározás is a klasszikus definícióra épül, és a következőképpen fogalmaz [39]: *„Az internetes spam több címre elküldött, lényegében azonos tartalmú kéretlen üzenet."* A szerzők részletesen megvilágítják a definíció egyes kifejezéseit. A *több* megvalósulásához elég a kettő, ellentétben az AOL internetszolgáltató meghatározásával, mely 24 óra alatt 25 vagy annál több azonos tartalmú levelet említ [25]. A *cím* itt nem csak emailcímet jelent, hanem tetszőleges végpontot, ahová üzenetet lehet küldeni (például hírcsoport), így a definíció magában foglalja az eletronikus üzenetek bármilyen formáját. A *lényegében azonos tartalmú* levelekben az üzenet az azonos, magát az üzenetet nem érintő változtatások (például személyfüggő megszólítás) nem vonják ki a levelet a definíció alól. A *kéretlen*ség fogalmát tekintették a legfontosabbnak: minden üzenet kéretlen, ha explicit módon nem kérte a címzett. Itt látják a legtöbb lehetőséget a kiskapukra, ezért jópár gyakran hivatkozott esetet felsorolnak, melyek *nem* jelentenek explicit kérést:

- egy honlap puszta meglátogatása nem jogosítja fel a honlap kezelőjét további kommunikációra;
- az eletronikus üzenet címzettje nem teheti levélküldő listára a feladót;

- emailcím közzététele nyilvános helyen (vagy egy űrlapon való megadása [25]) nem jogosít fel senkit arra, hogy a címre kéretlen levelet küldjön;
- nyilvános fórumok léte nem jogosít fel senkit arra, hogy oda tömeges üzenetet küldjön;
- egy témában való konkrét kérés nem jogosít fel egyéb témájú üzenetek küldésére;
- az explicit kérést mindenki csak saját maga teheti meg, azaz más címét nem szabad megadni, és a más által megadott címet nem szabad felhasználni.

Ezeket az eseteket összefoglalhatjuk úgy, hogy semmilyen külső forrásból szerzett címre nem etikus spamet küldeni [14].

A felhasználó szempontjából álláspontom szerint az a definíció lenne a legcélravezetőbb, hogy minden olyan üzenet spam, amire nem vagyok kíváncsi, aminek az olvasásával nem vagyok hajlandó az időmet tölteni. *Graham* spamszűrési módszere (ld. 5.1. fejezet), mint majd látni fogjuk, lehetőséget ad arra, hogy mindenki saját maga definiálja, hogy mit tart spamnek [12].

2.1.2 Az elnevezés eredete

A *SPAM* (Spiced Pork and Ham) eredetileg a *Hormel Foods* által gyártott löncshús konzerv neve. Az internetes világban használt jelentését egy *Monty Python*-jelenet nyomán kapta, melyből dolgozatom mottója is származik [42].

A jelenet [26] egy kávézóban játszódik, ahol az étlapon kínált minden étel spamet is tartalmaz. Hiába kéri a jelenetben szereplő házaspár szándékosan spam nélkül, a pincér mindig spammel együtt jegyzi fel az ételeket. A háttérben megjelenik a vikingek kórusa, énekük, a 'Spam, spam, csodás spam' egyre hangosabb lesz, végül elnyom minden értelmes kommunikációt.

A jelenet az internetes spam több fontos tulajdonságára rávilágít [37]:

- a spam lehetetlenné teszi a normális kommunikációt;
- a spam egy mindenütt jelen lévő, nemkívánatos dolog;
- a spam egy meglehetősen rossz minőségű „termék".

2.1.3 Az első spam

Eleinte a spam szót a MUD játékokban használták a romboló, ismétlődő üzenetek megjelölésére, aztán a sok hírcsoportba elküldött üzenetet hívták

így, később, miután elkezdték a Usenetet elárasztani reklámlevelekkel, a spamelés a nemkereskedelmi Usenet médium illetéktelen kereskedelmi célú felhasználását kezdte jelenteni. Ezután gyorsan jött az email spam [20, 42].

A legelső széles körben megismert spam egy 1994. áprilisában számos hírcsoportba elküldött reklám volt, melyben két ügyvéd – *Laurence Canter* és *Martha Siegel* – kínált „zöld kártya" ügyintézést az Egyesült Államokba településnek.

A spamelés története tehát körülbelül egy évtizedre tekint vissza. Érdekes, hogy az alapötlet, melyből a *Graham*-féle spamszűrő módszer (ld. 5.1. fejezet) is származtatható, már az 1994. márciusi [7] cikkben megjelenik, egy hónappal az első széles körben ismertté vált spam előtt.

2.1.4 A spamelés módszertana

A spameléshez két dolog kell: egy jókora emailcímlista, és valamilyen módszer az üzenetek célbajuttatására.

Címlisták esetében fontos elkülöníteni az ún. *opt-in* illetve *opt-out* listákat. Opt-in lista esetében kizárólag explicit kérésünkre kerülhetünk fel a listára, az opt-out listára ezzel szemben valahogy felkerül az ember, és csak azt kérheti, hogy vegyék le róla.

Az opt-in reklámlevelezés szigorúan elkülönítendő a lényegéből adódóan opt-out spameléstől. Jelenleg is működnek sikeres opt-in marketinggel foglalkozó internetes vállalkozások, például a `http://www.netcreations.com` [28].

A spamelők az emailcímek megszerzésének minden etikátlan módszerét alkalmazzák (ld. 2.1.1. fejezet): keresőrobotokkal böngészik az internetet és a Usenetet, begyűjtve minden @ karaktert tartalmazó karakterláncot, hamis indokokkal kérik az emailcím megadását, arra biztatnak, hogy add meg öt barátod címét is és akkor részt vehetsz a sorsoláson, stb. Másik módszerük, hogy megtippelik az emailcímeket, ezért például az `adam@aol.com` – ha létezik ez a cím – meglehetősen sok spamet kaphat [42].

Gyakran azt állítják, hogy a címlistákhoz legális úton jutottak hozzá, azzal érvelnek, hogy a birtokukban lévő listák opt-in listák, mivel le lehet róluk iratkozni. A fentiek fényében ez egyértelműen csúsztatás, ha a címzett előzőleg nem jelölt be egy – eredetileg nemre állított – jelölőnégyzetet, hogy kér levelet, akkor a címlista opt-out, és a küldött levél spam [12, 14].

Az üzenet célbajuttatásának a következő három módja van:

Spambarát szolgáltató Legegyszerűbb, ha az internetszolgáltató *spambarát*, azaz megengedi, hogy az általa rendelkezésre bocsátott címről leve-

lek millióit küldjék el. Úgy gondolhatjuk, hogy ilyen szolgáltató nem is létezik, mivel a szolgáltatók egyrészt költségeik csökkentése érdekében kordában akarják tartani a levélforgalom volumenét, másrészt nem szeretnének különféle spamellenes adatbázisokba bekerülni (ld. 3.2. fejezet), mert az ahhoz vezethet, hogy sokan domainjük területéről egyáltalán nem fogadnak emailt. Ezzel szemben a DesktopServer honlapján nagy méretű emailes kampányokhoz „megbízható", „profi", spambarát internetszolgáltatókat ajánlanak [45].

Open relay Az internetes levelezésben használt SMTP protokoll lehetővé teszi, hogy a címzett megadására szolgáló `RCPT TO:` sorban ne csak a szerver domainjébe tartozó, hanem tetszőleges cím szerepeljen. Ha a szerver nyitott (*open relay*), akkor az ilyen címekre is továbbítja a leveleket [28]. Az internet hőskorában ez volt az alapbeállítás, de a spamek elterjedtsége miatt manapság már követelmény, hogy egy levelezőszerver ne legyen nyitott. A spamellenes üzenet Kelet-Európába és Ázsiába lassabban jut el, viszont ezen területek internetes ellátottsága folyamatosan növekszik, így jellemző lett a koreai és kínai nyitott szerverek használata [20].

Az SMTP protokoll azt is lehetővé teszi, hogy a fejlécsorokat szinte tetszőlegesen töltsük ki, nyitott levelezőszerverek használatával a spamelő kiléte könnyen titokban tartható, hamis emailcím mögé rejthető.

Ezenkívül nyitott HTTP CONNECT-proxyk, nyitott SOCKS-proxyk és bizonyos nem biztonságos levelező CGI-scriptek is alkalmasak lehetnek a spam továbbítására [20].

Spamware A harmadik és egyben legkényelmesebb módszer az ún. *spamware*-ek használata. Ez a kifejezés a spamelés támogatása céljából készített szoftverek gyűjtőneve. Számos ilyen program létezik, ezek közül az egyik legrégebbi és talán legjelentősebb a *DesktopServer*.

Érdemes megnézni a program honlapját [45]. Végig úgy tesznek, mintha egy ártatlan és mindenki számára hasznos programot népszerűsítenének, gyanútlan emberben fel sem merül, hogy ilyen és ehhez hasonló szoftverek közreműködésével lepik el a kéretlen levelek az elektronikus postaládáját. A program 1996-ban jelent meg, két évvel az első ismert spam felbukkanása után (ld. 7. o.). A spamelés világméretűvé válását „emailforradalomnak", nevezik, és büszkén hirdetik, hogy ezt éppen a DesktopServer indította el. A Newsweek hetilap őket választotta meg a többszáz spamware közül a legjobbnak. (Érdekes, hogy egy ilyen tekintélyes újság milliónak napi bosszúságot okozó programok rangsorolásával foglalkozik.) A program „közvetlenül" – vélhetőleg egy beépített SMTP szerver által – képes kapcsolódni a címzettek levelesládájához, így általa kikerülhetők a szolgáltatók. Meglehetősen ellent-

mondásos, hogy bónuszként egy 57 millió emailcímből álló listát kínálnak, ugyanakkor figyelmeztetnek, hogy legyünk óvatosak, mert amelyik cég azt mondja, hogy az ő emaillistája friss, általában hazudik.

Paul Graham az említett hatalmas lista kapcsán megjegyzi, hogy lehetetlen, hogy az amerikai internetfelhasználók *fele* önkéntesen feliratkozott a DesktopServer listájára, azaz hazugság, hogy ez opt-in lista [15].

Az árulkodó jelek mellett van néhány konkrét bizonyítéka annak, hogy itt nem legális opt-in marketingről (ld. 8. o.), hanem spamelésről van szó:

- kész címlisták helyett a külön erre a célra kifejlesztett internetes emailcímeket gyűjtő programjukat ajánlják;
- „Az emailcímeket mind olyan internetfelhasználóktól gyűjtöttük, mint Ön. A hivatalos címeken kívül egy globális szűrőlistával kiszűrtük a panaszkodók legnagyobb részét is." [45];
- a DesktopServer „személyessé" tudja tenni a leveleket – például személyreszabott megszólítással – ezzel növelve a levél látszólagos hitelességét. Erre a trükkre opt-in lista esetén nyilvánvalóan nincs szükség.

Ez a bizonyos személyessé tétel visszafelé is elsülhet: ha 'Dear Joker!' megszólítással kapok egy levelet, azonnal tudom, hogy spam, mert ugyan `joker` az emailcímem első fele, de senki nem hív így.

2.1.5 A spamek tematikája

A spamek általában kétséges szolgáltatásokat kínálnak a gyógyfüvektől a szexuális szolgáltatásokon át a spamelésig. Egy tanulmány szerint a spam harmada gyors meggazdagodásra ad tippeket, negyede pedig pornó [20]. Joggal állapítja meg *Graham*, hogy általában olyasmiről szólnak, amiről jobban szerettük volna, ha azt se tudjuk, hogy létezik [14].

A következő témák szerepelnek leggyakrabban:

- pornóoldalak reklámja;
- Viagra;
- fogyókúrázási módszerek;
- hogyan legyél gyorsan és könnyen gazdag;
- spamware (ld. 2.1.4. fejezet).

Előfordul nem kereskedelmi jellegű, hanem vallási vagy politikai témájú spam is. Érdekességképpen lássuk ennek két szélsőséges példáját.

2002. márciusában a California kormányzói posztjára pályázó *Bill Jones* emberei egy koreai általános iskola open relay szervere segítségével küldték szét kampányhirdetéseiket [20].

2003. januárjában az Egyesült Államok hadserege indított spamhadjáratot Irakba azzal a céllal, hogy a rangidős iraki hivatalnokokat disszidálásra vegyék rá. A „Fontos információk" című levélben ezenkívül arra ösztönözték a katonai vezetőket, hogy háború esetén tagadják meg a tömegpusztító fegyverek használatát. Irak rögtön lépéseket tett a spamek blokkolására, mire az Egyesült Államok ráfanyalodott a rádión sugárzott üzenetekre illetve repülőgépről ledobott szórólapokra [22].

2.2 A spam, mint probléma

A spamelés eleinte néhány elszigetelt rosszindulatú site-hoz volt köthető, de a '90-es évek közepén gyorsan elterjedt és társadalmi problémává vált. Számos írás a mai internet egyik fő és egyre növekvő problémájának tekinti a spamelést, gondot jelent az internetszolgáltatók és a felhasználók szempontjából egyaránt. A felhasználók legfőbb bosszúsága az internettel kapcsolatban. A spam jelensége megmutatta, hogy a jelenlegi levelezőrendszer önmagában nem képes a szervezett visszaéléseket elhárítani. A rendeltetésszerű használat mellett végletesen alacsony költségekkel járó rendszerben néhány rosszindulatú felhasználó képes jelentősen növelni a költségeket [8, 20, 42]. 150 spamelő felelős a spamek 90 százalékáért [42].

A spam mennyiségéről szóló alábbi megállapítások, adatok jól mutatják, hogy nehéz túlbecsülni a probléma jelentőségét:

- a nagyobb internetszolgáltatók szerint a levelezőszerver-kapacitás körülbelül egyharmadát-kétharmadát a spam veszi el [42];
- a BrightMail hétmilliófféle spamet regisztrált 2003. áprilisában, ez 61%-os növekedés az egy évvel ezelőtti adathoz képest [8];
- 10 milliárd spamet küldenek naponta, 30 milliárd várható 2005-re [42];
- egy átlagos vállalatnál a bejövő levelek 10%-a spam [4];
- az Egyesült Államokban a levelek 37%-a, Angliában a levelek 8%-a spam [42];
- az egyik legelvetemültebb tömeges levélküldő egyhavi forgalma meghaladta a 134 Gbyte-ot [25].

2.2.1 Miért káros a spamelés?

A spamelés legfőképp azért elfogadhatatlan, mert a címzettre hárítja az üzenet költségeit, ezáltal lopásnak tekinthető [42].

Az internetszolgáltatók költségei növekednek, mert a spamelés pazarolja, megbénítja az erőforrásokat: a megnövekedett levélforgalom lefoglalja a sávszélességet, leterheli a levelezőszervereket, a célállomáson tárolni kell és fel kell dolgozni, ez a diszkek beteléséhez és a processzoridő fölösleges lefoglalásához vezet [25, 28]. Akár ún. *DoS* (Denial of Service) támadásnak is tekinthetjük egy szervezet számítástechnikai eszközei ellen [47].

A felhasználók költségei is növekednek: telefonos internetkapcsolat esetén növekszik a levelek letöltésének ideje, és jelentős időveszteségként jelenhet meg a spamek puszta letörölgetése. [1].

A spamekben gyakran előforduló pornográf tartalom korlátozás nélkül kerülhet gyerekek elé, ugyanis a spamelők nem mérlegelik, hogy az emailcím kié [1]. Ez számos országban jogszabályellenes.

Előfordul, hogy a napi megszokott spamtörölgetés folyamán fontos információ vész el, mert a felhasználó véletlenül egy rendes levelet hisz spamnek és letörli. Ez azért is valószínű, mert általában a feladó és a tárgy alapján dönt, mivel egyáltalán nincs kedve a spameket olvasgatni [47]. (Az ember spamfelismerő képességéről ld. még a 6.3.3. fejezetet.)

A fenti szempontokhoz *Graham* hozzáteszi, hogy a spamelés szétrombolja az internetes kommunikációt, és ez önmagában elég érv arra, hogy károsnak lehessen minősíteni. Sokszor érvelnek a spamelés mellett a szólásszabadság hangoztatásával, de ha egy megnyilatkozás egy bizonyos határon túl zavaró, akkor a szabad szólás sem védett többé [14].

A spamelés esetleg az internetes kommunikáció megszűnéséhez vezethet: az emberek a sok bosszankodás miatt egyszerűen nem fogják használni [25, 28].

Gunyorosan jegyzi meg egy levélíró a `news.admin.net-abuse.email` hírcsoportban, a spamellenes harc egyik fórumán: „A spamelők a televíziós reklámok egész bolygónkra kiterjedő hatékonyságát szeretnék elérni, de az email sajnos nem biztosítja ezt, az utálatos spamellenes aktivisták, az intoleráns internetszolgáltatók és a csalásról, a lopásról és a számítógépes bűncselekményekről szóló megszorító törvények miatt." [46]

Olyan érvelést is hallani, hogy a spam csak a rendes postai tömeges küldemények (pl. katalógusok, reklámok) elektronikus változata, tehát miért lenne etikátlan. Hogy még jobban megvilágítsuk a spamelés elfogadhatatlanságát, érdemes összehasonlítani a rendes postai kéretlen levelekkel [14, 42].

A postai küldemények jellemzői:

- a nyomtatás és a célbajuttatás költségeit a reklámozó fizeti, a darabszámmal arányos költségek miatt behatárolt a mennyisége;
- a kivitelezése igényes, és általában olyan dolgokat kínál – például ruházati cikkek –, amelyekre jó eséllyel szükségünk van;
- követhető, hogy ki küldi, számonkérhető, hogy mit;
- nem szakítja meg az aktuális tevékenységet, míg nem szánunk időt a napi posta kibontására.

A spam esetében mindennek épp az ellenkezője igaz:

- a költségek legnagyobb részét a címzett fizeti, a spamelő költsége nem arányos a darabszámmal, így mennyisége korlátlan lehet;
- igénytelen, általában szükségtelen, érdektelen termékeket kínál;
- a feladó általában hamis emailcím mögé rejtőzik, nem érhető utol, nem kérhető számon rajta a küldemény – adott esetben pornográf – tartalma;
- ha az email fontos munkaeszköz, minden spam bosszantóan félbeszakítja a munkát.

Paul Graham találó hasonlatával fejezem be az összehasonlítást: „Képzeljük el, hogy napi 100 katalógust kapunk, személyesen adják át mindet véletlenszerű időben érkező futárok, és mindegyik igénytelenül fénymásolt pornó, fogyókúrás és gyors meggazdagodási tippekről szóló reklámokat tartalmaz." [14]

2.2.2 A spam költsége

A spam nem ingyenes, a költség megjelenik, csak nem a spamelőt terheli. A spamelőre a költségek elenyésző része hárul, a többit a címzett, a károsult fizeti.

Egymillió spamet hozzávetőlegesen 250-500 $-ért (50-100 eFt-ért) lehet küldeni. Ez kétszázszor olcsóbb a postai tömeges küldeményeknél [47]. A DesktopServer hihetetlen arcátlansággal hirdeti, hogy a tömeges email szépsége az, hogy egyaránt használhatják szegények és gazdagok; a tömeges email világában mindenkinek egyenlő esélye van a milliomossá válásra; a tömeges email használatával egyáltalán nem kell bélyegekre és postaköltségre költeni többé [45].

Ha csak 2 másodpercet számítunk egy spam törlésére, akkor egymillió spam a címzettnek 2800 $ (600 eFt) költséget jelent [47]. *John Quaterman* szerint

csak a spamek kitörléséből származó időveszteséget számítva évi 87 millió $ (18 milliárd Ft) kár származik [28]. Amint a bevezetőben is említettem, egy Európai Uniós jelentés szerint a spamelésből származó összköltség eléri az évi 10 milliárd €-t (2500 milliárd Ft-ot) [9].

A spamelés létrejöttének és elterjedésének oka az alacsony költség, tehát a költségek kellő megemelésével a spamelés valószínűleg megszűnne (vö. 3.5. fejezet). Ha bevezetnék, hogy minden emailért szabott árat kell fizetni az SMS-ekhez hasonlóan, vélhetően egycsapásra megszűnne a spamelés, de ezt az ötletet eddig még senki nem vetette fel.

2.3 Spameléssel kapcsolatos perek

Ebben a fejezetben két, a közelmúltban lezajlott pert mutatok be, melyek nagyban befolyásolhatják a spamelés törvényi megítélését világszerte. Abba az irányba hathatnak, hogy a spamelést bűncselekményként meghatározó jogi szabályozások szülessenek.

2.3.1 Az EarthLink győzelme

2003-ban az Egyesült Államokban a bíróság megállapította, hogy az EarthLink nevű internetszolgáltató 16,4 millió $ (több, mint 3 milliárd Ft) kárt szenvedett a spamek miatt. Határozatot hozott *Howard Carmack* ellen, aki a tavalyi év során összesen 825 millió spamet küldő társaságnak volt a vezetője. A határozat megtiltja, hogy spamet küldjön vagy hogy bárkit hozzásegítsen ehhez [8]. *Carmack*ék hamis adatokkal és lopott hitelkártyaszámokkal hozták létre és folyamatosan változtatták az emailcímeket, ahonnan a spameket küldték, hogy ne lehessen rájuk találni. *Pete Wellborn*, az EarthLink ügyvédje szerint a határozat sokkal fontosabb, mint a kártérítés, mert megszabadítja az internetet ezektől az emberektől.

Az Egyesült Államokban jópár ilyen per folyt a közelmúltban, a kiterjedt ügyfélkörrel rendelkező AOL internetszolgáltató huszonöt ehhez hasonló pert nyert már meg [8].

2.3.2 A T3 Direct – *Joey McNicol* per

Furcsább eset játszódott le Ausztráliában: a spamelő perelte be az áldozatot.

*Joey McNicol*nak elege lett a T3 Direct nevű kifejezetten spameléssel foglalkozó marketing cégtől kapott spamekből, közzé tette a cég adatait a Useneten, nem sokkal később a T3 Direct IP-címe megjelent a SPEWS-on. Ez a cég teljes emailforgalmának megbénulásához, és jelentős bevételkieséshez vezetett.

Ennek kapcsán a T3 Direct perbe fogta *McNicol*t, és 44000 $ (9 millió Ft) kártérítést követelt tőle [40].

McNicol és támogatói köré mozgalom szerveződött. „*Joey McNicol* Dávid-Góliát harcot vív egy perthi direkt marketing céggel, ami a spam jövőjét döntheti el." [33]

2002. októberében végül döntött a bíróság, a T3 Direct keresetét elutasították. A cég eredeti panasza az volt, hogy *McNicol* megalapozatlanul állította, hogy a cég spamet küld. A per során a T3 Direct képviselője nem tagadta, hogy tényleg küldenek tömeges emailt rendes üzletmenetük folyamán, így a bíróság logikusan döntött amellett, hogy *McNicol* állítása egyáltalán nem volt megalapozatlan.

3 A spam elleni küzdelem

A spam elleni küzdelem két nagy színtéren folyik, mindkettőnek a célja a spamelés megszüntetése. Az egyik a spamelés törvényi elítéléséért folyó küzdelem, a másik pedig a mindennapos harc a spamlevelek ellen, azok blokkolása illetve szűrése útján [42]. A kétféle megközelítés kiegészíti egymást.

3.1 Spamellenes szervezetek

A több éve létező CAUCE (Coalition Against Unsolicited Commercial Email) nevű szervezet a jog eszközeivel kíván fellépni; tevékenységének eredménye, hogy számos USA-tagállam, és néhány európai állam is spamelést tiltó törvényt hozott. Újabban különböző EU-direktívák is ezt követelik meg az Európai Unió tagjaitól [20]. Az amerikai spamellenes törvénytervezet abból indult ki, hogy a spam olyan káros, mint a kéretlen reklámfax. Az ilyet már évek óta küldeményenként 500-1500 $ (100-300 eFt) pénzbírsággal sújthatják az Egyesült Államokban, amit a küldő a címzettnek (!) köteles fizetni. E törvény következtében a reklámfax gyakorlatilag megszűnt [28]. *Péterfalvi Attila* adatvédelmi biztos ajánlása szerint az emailcím személyes adat, azaz senki nem használhatja fel illetéktelenül. Következésképpen a spamelés Magyarországon is jogsértő, illetve az lesz EU-csatlakozásunk pillanatától, amikortól az adatvédelmi biztos ajánlásai kötelező érvényt kapnak [43].

A törvényi tiltás általában az UCE-re irányul, ahogy ez a CAUCE nevében is benne van, ugyanakkor a spam egy jelentős része nem kereskedelmi – például politikai, vallási – témájú, viszont éppen olyan zavaró [42]. Célszerű lenne tehát, ha a születendő jogi szabályozás a 6. oldalon megfogalmazott definícióhoz hasonló átfogó definíciót alkalmazna.

Sajnálatos, hogy a spamelőket meglehetősen ritkán perelik be, és a bíróságok ilyen esetekben is csak ritkán hoznak rájuk nézve elmarasztaló ítéletet [20]. Ezügyben napjainkban biztató fejlődésre van remény (ld. 2.3.1. fejezet).

2003-ban alakult meg az ASRG (Anti-Spam Research Group), mert az internet működésével és fejlődésével foglalkozó szervezetek már nem bírják tétlenül szemlélni az ártatlan internetezők és a gátlástalan hirdetők között dúló spamháborút [18]. Az új szervezet szerint is először egy szabatos definíció kell. *Jason Catlett* újszerű ötlete szerint létre lehetne hozni egy technikai szabványt a spam SMTP alapú letiltására, a „Nem kérünk reklámot!" postaládai felirat elektronikus megfelelőjeként.

3.2 Blokkolás

Blokkolásnak nevezzük azt a tevékenységet, amikor bizonyos forrásokból eleve nem fogadunk emailt, mert alapos a gyanúnk, hogy főleg spamet kapnánk. Ez a megközelítés általában a rendes levelek elvesztésének nagyobb kockázatával jár, viszont nagymértékben csökkenti a spamforgalmat, ezért az internetszolgáltatók kedvelt eszköze.

Több olyan folyamatosan naprakészen tartott feketelista létezik az interneten, ahol a spambarát internetszolgáltatókat, azon site-okat, ahonnan a spamek érkeznek, spamelők emailcímeit, azaz azokat a forrásokat tartják nyilván, ahonnan jó eséllyel várhatunk spamet.

A DNSBL (DNS-based Blackhole List) egy speciális DNS-zóna, amely spamelő vagy spamelést támogató site-ok IP-címeit sorolja fel [20].

A legrégebbi ilyen szolgáltatás a MAPS-RBL (Mail Abuse Prevention System – Realtime Blackhole Lists).

Az ORDB (Open Relay Database) az open relay-ket listázza, folyamatosan teszteli a levelezőszervereket, és ha nyitott szervert talál, akkor felveszi a listára, egyben értesíti a szerver rendszergazdáját, hogy szüntesse meg az open relay-konfigurációt. Ehhez segítséget is adnak, és ha a rendszergazda megtette a megfelelő lépéseket, a szervere lekerül a listáról. 1999-ben középiskolai rendszergazdaként én is végigjátszottam ezt a folyamatot.

A SPEWS (Spam Prevention Early Warning System) egy névtelenül üzemeltetett DNSBL. A 'spews' szó egyébként azt jelenti, hogy 'kiköpi', amint tényleg ezt is teszi a rendszer a spambarát site-okkal. Paradox módon az oroszországi Irkutszkban működik, hogy védve legyen a DNSBL-eket folyamatosan zaklató nyugati perektől. Egyesek szerint a SPEWS túl sok fals pozitív – hibásan spamnek vélt rendes levél – forrása, ezzel szemben az is tapasztalat, hogy a SPEWS-on kiírt szerverekről szinte kizárólag spam érkezik [20].

A DNSBL-ek használata a következőképpen történik: le lehet kérdezni, hogy egy adott IP-cím szerepel-e a listán. Ha például egy levelezőszerverről, ahonnan leveleket kap a szerverünk, kiderül, hogy szerepel a listán, dönthetünk úgy, hogy egyáltalán nem fogadunk tőle levelet.

A feketelisták eredeti szerepe nem a spamblokkolás volt, hanem a spambarát site-ok pellengérre állítása, rászorítva ezzel őket a spamellenes intézkedések megtételére. Ma mégis sok esetben azt az egyszerű eljárást választják, hogy a DNSBL-en szereplő domain területéről egyáltalán nem fogadnak emaileket, azaz a DNSBL-ek spamblokkolásban játszott szerepe megnőtt.

Egyes vélemények szerint a feketelisták használata olyan, mint amikor ágyúval lövünk verébre [47]. A fenti módszer *Graham* szerint is nagyon rossz

hatásfokkal működik. Egy tanulmányra hivatkozva mondja, hogy a MAPS-RBL, ami talán a legjobb a feketelisták között a spameknek mindössze 24%-át fogja meg, ugyanakkor a rendes levelek 34%-át spamnek véli. A feketelisták – eredeti céljuknak megfelelően – talán hathatnak úgy a szolgáltatókra, hogy fejezzék be a spamelés támogatását, viszont fennáll annak a veszélye, hogy egyesek haragosaikat írják fel a listára. „Gondolkodj globálisan, cselekedj lokálisan!", azaz feketelisták helyett alkalmazz szűrést - véli *Graham*. Ez persze nem csökkenti a spamforgalmat, így a szolgáltatók költségeit sem, de ha reményei beigazolódnak és a statisztikai szűrés kapcsán a spamelés teljesen megszűnik, akkor idővel a szolgáltatók gondja is megoldódik [11].

A feketelistákhoz tartozik még két érdekes megközelítés [20].

Az egyik a *Vernon Shryver* által üzemeltetett DCC (Distributed Checksum Clearinghouse). Itt a bejövő emailekből a megadott algoritmussal egy ellenőrző összeget képezünk, ezt az értéket küldjük el a szervernek, a szerver válaszként megadja, hogy hányszor találkozott már ezzel az értékkel. A rendszer ún. *fuzzy checksum*okat használ, melyeknek az igazi ellenőrző összegekkel szemben pont az a lényege, hogy a levél kis megváltozására csak kicsit változzanak meg. Ezzel a módszerrel a levél tömeges voltát tudjuk mérni.

A másik *Vipul Ved Prakash* Razor nevű rendszere. Itt bárki elküldheti a saját spamjeit a szerverre, a rendszer felhasználói bejelentésen alapul. A szerver a komplett spameket tárolja hashelve. A blokkolás természetesen úgy valósítható meg, hogy megnézzük, hogy a beérkező levél szerepel-e a Razor adatbázisában.

A feketelisták pontosságát növelni lehet ún. *fehérlisták* használatával. A fehérlista olyan forrásokat tartalmaz, ahonnan mindenképpen elfogadjuk a leveleket. Nyilvántarthatjuk például levelezőpartnereink címlistáját, megtehetjük, hogy csak akkor fogadunk el egy levelet, ha egy előre meghatározott jelszó szerepel a tárgysorban.

Dolgozhatunk kizárólag fehérlisták alapján is, úgy, hogy a fehérlistán nem szereplők leveleit egyszerűen visszaküldjük. Ennek fejlettebb változata, mikor lehetőséget adunk a fehérlistára való felkerülésre, például, hogy egy egyszerű, de gyakran változtatott rejtvényt kell megoldani, és a megoldást bele kell tenni a tárgysorba [1, 47]. Ezzel elkerüljük, hogy automata levelek felkerülhessenek a fehérlistára, így kiszűrjük a spamek jelentős részét, mivel az automataság a spamek egyik legfontosabb tulajdonsága (ld. 6. o.).

A fehérlisták a szűrők kiegészítőjeként is használhatók, ezekről lesz szó a következő fejezetben.

3.3 Szűrés

Szűrésnek nevezzük azt a tevékenységet, mikor a beérkező leveleket a tartalmuk alapján spamként vagy rendes levélként jelöljük meg. A szűrés felhasználóbarátabb módszer, mint a blokkolás, ugyanis kevésbé jár a rendes levelek elvesztésének kockázatával. Ha a spameket nem töröljük, hanem csak külön mappába tesszük, a kockázat minimális. Ugyanakkor a spamforgalmat nem csökkenti, ezért a szolgáltatók spam által okozott költségeit sem. *Graham* szerint a blokkolás szűk látókörű, mert csak a napi gondot akarja megoldani, a spamelés káros hatásait akarja elkerülni, a szűrés viszont távlatokban gondolkozik, mert célja a spamelés teljes megszüntetése [11, 15].

Legkezdetlegesebb módszer, hogy egy feketelistát vezetünk a spamgyanús kifejezésekről [28], és ha ezek közül valamelyik szerepel egy levélben, akkor azt spamnek tekintjük. A régebbi – elsőgenerációs – szűrőprogramok ilyen listákkal dolgoztak, később szabályok formájában bizonyos spamekre jellemző tulajdonságokat fogalmaztak meg, és ezek meglétét vizsgálták, ezt nevezzük heurisztikus szűrésnek. Fontossá vált az emberi erőforrás-igény csökkentése a szűrők fejlesztésében, azaz, hogy ne kelljen spamek százait végigolvasni, és ezek alapján kitalálni különféle spamtulajdonságokat leíró szabályokat. A másodikgenerációs statisztikai szűrők biztosítják ezt az automatizáltságot.

Meg kell jegyezni, hogy minden szűrős megoldás szükségképpen néha spamnek ítél rendes levelet, ezáltal fontos információ elvesztéséhez vezethet. Ezt nevezzük *fals pozitív* találatnak. A minimális fals pozitív arány a spamszűrők legfontosabb minőségi követelménye. A szűrők teljesítményének valódi mértéke nem önmagában az, hogy a spamek mekkora részét fogják meg, hanem, hogy a spamek mekkora részét fogják meg úgy, hogy közben közel nulla a fals pozitív találatok aránya [11].

3.3.1 Szerverszint vagy felhasználói szint

Általában szerverszinten blokkolást, felhasználói szinten szűrést használnak, de elvben mindkét módszer alkalmazható mindkét szinten.

A szerverszintű szűrés előnye, hogy a spamek még ideiglenesen sem foglalják le az erőforrásokat [28], viszont a fals pozitívok veszélye miatt ezt nehéz megvalósítani [20], ezenkívül felveti a levéltitok megsértésének problémáját is (vö. 3.6. fejezet), ugyanis hogyan lehet szerverszinten szűrni az egyes felhasználók leveleit anélkül, hogy a szűrő „beleolvasna”.

Felhasználói szinten árnyaltabb, személyreszabottabb szűrést lehet megvalósítani. Azok a szűrők tűnnek hatékonyabbnak, melyek az egyes felhasználók saját emailforgalmára épülnek. A fő előnyük, hogy mind különbözőek [10].

Annál jobb, minél többféle szűrő van, mivel így a spamelőknek sokkal nagyobb erőfeszítésbe kerül az összeset, vagy legalábbis sokat kicselezni [47]. Ideális esetben a szűrő mindenkinek a saját leveleire épül, a szoftver külön tárolja a rendes és a spamleveleket, és ezek alapján automatikusan fejleszti a tudását és a teljesítményét [12].

3.3.2 Az első generáció – szabályok

Az elsőgenerációs szűrők szabályokat használtak a spamjellegzetességek felismerésére [15].

Az ilyen szűrők tudásbázisának építésénél nagymértékben támaszkodnak az emberi erőforrásokra, szakértők munkájára. A szabályokat állandóan fejleszteni kell, hogy folyamatosan alkalmazkodjanak az épp divatos spamekhez. A statikus szabályalapú szűrés nehezen karbantartható és hajlamos a fals pozitív típusú hibára. A folyamatos fejlesztés, karbantartás hibalehetőségeket is hordoz magában. Az elsőgenerációs szűrők nem személyreszabottak, a tudásbázisuk statikus, központilag meghatározott [38, 42].

Egy automatikusan adaptálódó, tudásbázisát automatikusan fejlesztő szűrő kiküszöbölné ezeket a problémákat [1, 47].

Az első generáció egyik fejlett, heurisztikus szűrést alkalmazó példája a *SpamAssassin* nevű program [41].

Tudásbázisában több, mint 400 szabály található. Ezek reguláris kifejezések, melyeket a program a bejövő levelekre alkalmaz, ennek eredményeként az email bizonyos pontszámokat kap. Ha az összpontszám túllép egy határt, akkor a levél spamnek minősül. A program a szűrésen kívül számos egyéb módszer használatára konfigurálható. Ilyenek a különböző DNSBL-ek, a DCC vagy a Vipul's Razor [20].

A szabályok fejlesztése, a pontszámok meghatározása a SpamAssassin esetében is szakértelmet kíván, a *Paul Graham* által javasolt statisztikus szűrés módszere viszont nem igényel emberi beavatkozást.

3.3.3 A második generáció – statisztikai szűrés

Az elmúlt két év vízválasztó volt a spam elleni küzdelemben [20]. Ebben az időszakban jelentek meg az új típusú statisztikai szűrők. Ezek az emailekben lévő szavak gyakoriságára épülnek, automatikusan építhető tudásbázisukban minden előforduló szóhoz tartozik egy valószínűség, ami megadja, hogy a szó inkább spamekben vagy inkább rendes levelekben fordul elő. Ezek alapján osztja a szűrő a bejövő leveleket két csoportra.

Ezeket a szűrőket *Paul Graham* 'A Plan for Spam' [12] című dolgozata hozta

be a köztudatba, ugyanakkor két hasonló elven működő program, az *ifile* [34] és a *CRM114 Discriminator* [47] elérhető volt évek óta.

Úgy tűnik, hogy a másodikgenerációs statisztikai alapon működő szűrők minőségileg jobban teljesítenek elődeiknél. Általában mondható, hogy a spamek 99 százalékát elkapják, megközelítőleg nulla fals pozitív mellett. A döntéshez gyakran naív bayesi osztályozót (NBC-t), vagy valamilyen ehhez hasonló eszközt használnak.

Számos előnyös tulajdonsággal bírnak [15]:

- A spamek döntő részét elkapják, a lefedettség (ld. 29. o.) $> 0{,}99$.
- Nagyon kevés a fals pozitív találat, a pontosság (ld. 29. o.) ≈ 1.
- A helyesen osztályozott spamekkel és rendes levelekkel időről időre betanítható, így folyamatosan adaptálódik.
- Lehetővé teszi, hogy mindenki maga definiálja, hogy mit tart spamnek.
- Nehéz kicselezni őket. A spemmelő egyetlen lehetősége, hogy a szókincsét olyanná alakítja, mint a címzett rendes leveleinek szókincse. Ezt kétféleképpen érheti el: vagy kevesebb spamgyanús szót, vagy több ártatlan szót használ. Az első esetben bármilyen szavakkal fogalmaz, pontosan ezek az általa használt szavak lesznek a spamgyanúsak, a második módszert pedig nem tudja megvalósítani, mivel nem ismeri az egyes felhasználók személyes levelezésében gyakran használt szavakat. Ez is egy érv amellett, hogy érdemes felhasználói szinten szűrni [13].

Graham módszerének leírása az 5.1., alkalmazása a 6. fejezetben található.

3.4 Tanácsok

Ne dőljünk be olyan nézeteknek, melyek szerint a spam a „szabad kommunikáció", az „elektromos marketing", a „modern internet" normális velejárója [28].

Az internetszolgáltatók szabályzatukban rögzítsék a spamelés tilalmát, lépjenek fel a spamelés ellen, zárják le az open relay-ket, legyenek elérhetők a szabványos `abuse@domainnév` címen.

Semmiképpen ne válaszoljunk a spamekre. Sok spam felajánlja, hogy leiratkozhatunk a listáról, ezt sose tegyük [14, 42]. A spamelő a válaszból ahhoz az értékes információhoz jut hozzá, hogy az emailcím létezik, olvassák, így érdemes még több spamet küldeni rá. Még egy érv lehet a *reverse spam* esete, amikor a spamelők olyan valakinek az emailcímét teszik a feladó helyére, akinek ártani akarnak, az ártatlan áldozat meg csak olvashatja sorra

a tiltakozó leveleket [28]. Létezett az Egyesült Államokban egy szervezet, az IEMCC (Internet EMail Marketing Council). Ez azt ígérte, hogy ha valaki leiratkozó levelet küld egy megadott címre, akkor nem kap több spamet. A tapasztalat azt mutatta, hogy a leiratkozó levelet követően még több spam érkezett. Később meg is szűnt ez a szervezet. Korábban azt tanácsolták a felhasználóknak, hogy bátran iratkozzanak le, ha nem akarnak több spamet kapni, mára bebizonyosodott, hogy a leiratkozó címek nagy része csapda.

Válasszunk nehezen kitalálható emailcímet [42], és/vagy használjunk ún. *álcímeket*, mikor saját üzeneteink feladóját adjuk meg [28]. Álcímek esetében kicsit megváltoztatjuk az emailcímet, és mindig mellékeljük a leírást, hogy hogyan lehet az álcímből az igazit kinyerni. Például ha üzenetem feladójaként a `joker-nem-kerek-spamet@ludens.elte.hu` szerepel, akkor mondjuk az aláírásban közlöm, hogy válaszoláskor távolítsa el a `-nem-kerek-spamet` karaktersorozatot az emailcímből. A domainnevek megváltoztatása nem javasolt, mert számos domain amiatt vált használhatatlanná, hogy gyakran használják álcímekben. Ha éppen csak emailcímekről van szó a levelezésünkben, akkor jobb a @ karaktert teljesen eltüntetni, például `(a)`-val helyettesíteni.

Jelentsük be a spameket a DCC-nél, a Vipul's Razornál [42]. Tegyünk panaszt a küldő rendszer rendszergazdájánál, használjuk az 2142-es RFC-ben szabványosított `abuse@domainnév` címet [5]. Ez a legbiztosabb kapcsolatfelvételi hely a spameléssel és más visszaélésekkel kapcsolatos problémák tisztázása céljából. Kipróbáltam az `abuse@yahoo.com` és az `abuse@excite.com` címeket. Mindkét helyen nagyon készségesek voltak, és megállapították, hogy a spamekbe, amiket a panaszomban küldtem nekik, csak belehamisították az ő domainnevüket, ami ellen sajnos nem tudnak semmit tenni, emellett biztosítottak afelől, hogy mindent megtesznek a spamelés ellen.

Legalább annyit állítsunk be, hogy a levelezőprogram ne mutassa automatikusan a képeket, illetve ne irányítsa a felhasználót automatikusan a levélbeli honlapra [42], így időt spórolhatunk és elkerülhetjük, hogy pornográf tartalom kerüljön gyerekek elé.

3.5 Esélyeink a spamelőkkel szemben

Gyakori az a vélekedés, hogy a spamelők mindig egy lépéssel a spamellenes aktivisták előtt járnak, ezért a küzdelem az idők végezetéig fog tartani. A spamkészítők mindig újabb és egyre trükkösebb módjait találják ki leveleik célbajuttatásának [28]. A cégek szűrőket fejlesztenek ki, de a spamelők eddig még mindig megtalálták a módját, hogy hogyan cselezzék ki ezeket az óvintézkedéseket [8].

Graham ezzel szemben annak a reményének ad hangot, hogy ha kellően

hatékony szűrőket fogunk széleskörűen alkalmazni, akkor meg fog szűnni a spamelés. A spamelők *Achilles*-sarka maga az üzenet. Bármilyen eléjük állított akadályon átküzdik magukat, de az üzenet tartalmáról nem mondhatnak le [12].

Kellően hatékony szűrő az a szűrő, amely olyan magas lefedettséget biztosít, hogy a válaszolási ráta csökkentése révén a spamelők költségeit a postai tömeges küldemények szintjére vagy afölé emeli, ezáltal a spamelés nem éri meg a továbbiakban.

Graham szerint a bayesi spamszűrés lesz a valóban leghatékonyabb [15].

A spam körülbelül kétszázszor olcsóbb, mint a postai tömeges küldemény, így 99,5 százalékos lefedettségű szűrő kell ahhoz, hogy a válaszolási rátát kétszázadára csökkentsük, azaz a spamelők költségét kétszászorosára, a postai küldeményekkel azonos költségszintre növeljük. Ezt a szintet a CRM114 Discriminator biztosítja [47].

3.6 Spamszűrés és magánszféra

Sok internetszolgáltató alkalmaz spamszűrést, és a felhasználók spamjeit külön mappában tárolja. Ha elterjed, hogy lehetséges kiszűrni a spam nagy részét, a nagy szolgáltatók felhasználói valószínűleg kötelezve lesznek a spamszűrő használatára [13]. Egyes szervezetek nemtetszésüket fejezik ki, hogy némely szolgáltató a felhasználók tudta nélkül használ spamszűrést, kockáztatva ezzel, hogy egy-két rendes levél is elvész [42].

Ha spam érkezik, a legrosszabb ami történhet, hogy a felhasználó válaszol rá, ez ugyanis megerősíti a spamelőt abban, hogy érdemes „dolgoznia", így további spamek áradatát indítja el. Az internetszolgáltatók félve ettől hajlamosak különféle trükköket alkalmazni, hogy elvegyék a felhasználók kedvét a spammappák böngészésétől, például zagyvasággal töltik fel a spamek tárgysorát [15].

Mindez persze csak akkor jelent gondot, ha véletlenül egy rendes levél került a spamek közé, ami a legjobb szűrő esetén is előfordulhat. *Péterfalvi Attila* adatvédelmi biztos ajánlása szerint az email tartalma levéltitkot képez, ahhoz hozzáférni a címzetten kívül senkinek sincs joga [43]. Ez kérdésessé teszi a szerverszintű spamszűrés létjogosultságát, mert a szűrő „beleolvas" a levélbe. Szerverszinten csak a blokkolás korrekt és az is csak akkor, ha ismert a felhasználók előtt, hogy pontosan mit blokkolnak.

4 Szövegosztályozás

A spamszűrés tekinthető olyan szövegosztályozási problémának, melyben összesen két kategória van: a rendes levelek és a spamek. Ebben a fejezetben a naív bayesi osztályozót (NBC: Naïve Bayesian Classifier) járom körül, jellegzetességeit, a hozzá kapcsolódó módszereket, melyekből *Graham* ötleteket merített az 5.1. fejezetben ismertetett módszeréhez. Szó esik az osztályozó markovi kiterjesztéséről is.

A programnak meg kell „értenie" a szöveget ahhoz, hogy osztályozni tudja [44]. A nyelv megértése nem szükséges a nyelv felismeréséhez [7].

A fenti, egymásnak ellentmondó állításpár utóbbi tagja a szimpatikusabb számomra, ez ad ugyanis reményt arra, hogy kizárólag statisztikai alapon, mindenféle nyelvi ismeret nélkül, sőt nyelvektől függetlenül eldönthessük, hogy egy levél spam vagy nem spam.

4.1 A klasszikus naív bayesi osztályozó – az NBC

Az NBC az alábbiak szerint működik [29].

Jelöljük v_j-vel az osztályozandó szöveg K darab tulajdonságát – általában azt, hogy a j-edik szó szerepel-e a szövegben –, ezeket foglaljuk össze a d vektorban. C legyen a kategóriák halmaza, $c \in C$ pedig egy kategória.

A *Bayes*-tétel alapján:

$$P(c|d) = \frac{P(c) \cdot P(d|c)}{P(d)}$$

feltéve, hogy az egyes v_j tulajdonságok függetlenek egymástól a

$$P(c|d) = \frac{P(c) \cdot \prod_{j=1}^{K} P(v_j|c)}{P(d)}$$

képletet kapjuk. Ez alapján a MAP (maximum a posteriori) osztályozó – mivel $P(d)$ állandó – a következő:

$$c^* = \arg\max_{c \in C}\{P(c|d)\} = \arg\max_{c \in C}\{P(c) \cdot \prod_{j=1}^{K} P(v_j|c)\}$$

Ha nem ismerjük a prior eloszlást, $P(c)$-t, akkor általában szokás egyenletesnek venni, így kapjuk meg az ML (maximum likelihood) osztályozót:

$$c^* = \arg\max_{c \in C}\{\prod_{j=1}^{K} P(v_j|c)\} = \arg\max_{c \in C}\{P(d|c)\}$$

Ha a $P(v_j|c)$-ket, azaz a tulajdonságok valószínűségét az egyes kategóriákon belül ismerjük, ezeket behelyettesítve megkapjuk c^*-ot, azt az osztályt, melybe a d-vel jellemzett szöveg tartozik.

Alapesetben felügyelt tanulásos keretrendszer szolgál az osztályozó tanítására, az egyes osztályokba tartozó példák alapján becsüljük a $P(v_j|c)$ paramétereket. Az egy osztályhoz tartozó példák összességét, nyelvészeti kifejezéssel *korpusz*nak nevezzük.

A dokumentum leírására általában a leghatékonyabbnak bizonyult ún. *polinomiális eseménymodell*t alkalmazzák [24]. Ez tekintetbe veszi a szógyakoriságokat, viszont nem veszi tekintetbe, hogy mely szavak nem fordulnak elő, az egyes szóelőfordulásokat eseményeknek veszi, és a dokumentum ilyen eseményeknek a gyűjteménye. Úgy tekint az N darab K-féle szóból álló szövegre, mintha N-szer végeztek volna el egy K-féle kimenetellel rendelkező kísérletet, azaz egy dokumentum adott osztályra vonatkozó valószínűsége K-adrendű polinomiális eloszlásból számítható ki [2, 100. o.].

A $P(v_j|c)$ paramétereket a

$$P(v_j|c) = \frac{N_j^c}{N^c}$$

képlet alapján becsüljük, ahol N_j^c jelöli a v_j előfordulásainak számát a c kategóriában, N^c pedig a szavak számát, azaz $N^c = \sum_j N_j^c$. Ez szemléletesen annyit jelent, hogy a c kategória esetén a v_j szó valószínűségét a gyakoriságával becsüljük.

Korábbi tapasztalatok szerint az NBC szövegosztályozóként nagyon jól működik, egyszerű, és nincs benne nyelvfüggő vagy egyéb bedrótozott elem.

Azonban már az 1998-as, szövegosztályozást spamszűrésre alkalmazó cikk felhívja a figyelmet, hogy a spamszűrés esetében a sima NBC nem kielégítő, kell valamilyen módszer a fals pozitívok elkerülésére [38].

Az NBC fontos elemei:

- a *függetlenségi feltételezés*;
- a *prior eloszlás*;
- az ún. *tisztítás*, azaz, hogy mely tulajdonságokat hagyjuk el eleve;
- a *tulajdonságkiválasztás*, azaz, hogy melyik K darab tulajdonságot vegyük figyelembe végül a döntésnél;
- és az ún. *simítás*, ami a tanítómintában elő nem forduló szavak kezelését jelenti.

Ezeket tekintjük át részletesebben.

4.1.1 Függetlenségi feltételezés

Feltettük, hogy a szövegben az egyes szavak előfordulása független egymástól, azaz

$$P(d|c) = \prod_{j=1}^{K} P(v_j|c)$$

E feltételezés miatt kapta az osztályozó a „naív" elnevezést.

A függetlenségi követelmény, amire az NBC-k épülnek, szinte sosem teljesül természetes adathalmazokra, legfőképp nem szövegekre. Ez irányította a kutatókat a függetlenségi követelmény lazítása/enyhítése felé [21]. Éppen a függetlenségi követelmény lazítási szándéka vezetett az NBC markovi kiterjesztéséhez (ld. 4.2. fejezet).

Azért vagyunk kénytelenek feltételezni a függetlenséget, mert képtelenség a $P(d|c)$-ket közvetlenül becsülni, a d rengeteg lehetséges értéke miatt [1]. A feltételezés a legtöbb esetben helytelen, de nagymértékben megkönnyíti a számításokat [23]. Az NBC meglepően hatékony és közel optimális hibával dolgozik, annak ellenére, hogy a függetlenségi követelményt általában súlyosan megsértik [6].

Spamszűrés esetében nagyon határozottan nem igaz a függetlenség [47], de mint látni fogjuk, itt is meglehetősen jó eredménnyel működik az NBC.

4.1.2 Prior eloszlás

Ismeretlen prior eloszlás esetén általában az egyenletes eloszlás alkalmazását javasolják. Fontos eredmény, hogy egyenletes prior eloszlással a naív bayesi osztályozó minimalizálja a hibát [7, 29].

4.1.3 Tisztítás és tulajdonságkiválasztás

Ennek a lépésnek a lényege a dimenziószám csökkentése, azaz, hogy ne kelljen túl sok adattal számolni.

A tisztítás általában a tanítómintában túl kevésszer előforduló szavak elhagyását jelenti [38], de előfordult, hogy a kötőszavakat és a minden kategóriában azonos valószínűségű szavakat hagyták el [44], vagy éppen szótővé alakítás után a 100 leggyakoribb angol szót [1].

A tulajdonságkiválasztás során eldől, hogy végülis mely tulajdonságok fogják a d vektort alkotni.

Spamszűrés esetén előfordult, hogy a leveleknek csak a tárgysorát és a törzsét tekintetbe véve az 500 legfontosabb szót választották ki [38], egy másik eset-

ben pedig a csatolt fájlok és a HTML-kód elhagyása után választottak ki bizonyos szavakat [1], a *SpamBayes* projektben a tisztítás során a HTML-kódon kívül bizonyos fejlécsorokat is elhagytak [32].

4.1.4 Simítás

A paraméterek becslése a

$$P(v_j|c) = \frac{N_j^c}{N^c}$$

képlet alapján történik. Fontos kérdés, hogy mit kezdjünk a tanítómintában elő nem forduló szavakkal, ugyanis esetükben $N_j^c = 0$ miatt $P(v_j|c) = 0$ lenne, ezáltal $\prod_{j=1}^{K} P(v_j|c)$ is nulla lenne, így ezt a kategóriát az osztályozó mindenképpen elvetné.

Az egyik elterjedt módszer az ún. *Laplace-simítás* egyszerűsített változata, az ún. *plusz-egy-simítás*, mikor a fenti becslés helyett a

$$P(v_j|c) = \frac{N_j^c + 1}{N^c + K}$$

becslést alkalmazzák, ami által kiküszöbölik a nulla valószínűségeket.

Még primitívebb technika, hogy ha nulla lenne egy valószínűség, akkor helyette mondjuk $\varepsilon = 0{,}01$-et veszünk [44].

A *Laplace*-simítás levezetését a [7] cikkben találjuk meg, egyéb simítási technikákról pedig a [29] cikkben olvashatunk.

4.2 Markovi kiterjesztés

Régóta világos volt, hogy a bayesi módszereket fel lehet használni *Markov*-processzek által készített nyelvi modellek kezelésére [7].

A *Markov*-lánccal bővített NBC-t, az egyszerű NBC és az n-gram modellezés, összekombinálásával hozták létre [29]. Az n-gram egyszerűen egy szó-n-est, azaz egymás után következő n darab szót jelent. Az n-gram modell markovi függőséget ad meg a szavak között, ami sokkal közelebb áll a valósághoz, mint a teljes függetlenség. Működése a következő:

Jelölje w_i a T darab szóból álló szöveg i-edik szavát.

A dokumentum valószínűsége:

$$P(w_1 \dots w_T)$$

a feltételes valószínűségdefiníciója alapján,

$$\prod_{i=1}^{T} P(w_i|w_1 \dots w_{i-1})$$

feltéve az n-edrendű markovi tulajdonságot, hogy az i-edik szó csak a megelőző n darab szótól függ,

$$\prod_{i=1}^{T} P(w_i|w_{i-n+1}\dots w_{i-1}).$$

A szorzat tényezőit becsülhetjük így:

$$P(w_i|w_i - n + 1\dots w_{i-1}) = \frac{N_{w_{i-n+1}\dots w_i}}{N_{w_{i-n+1}\dots w_{i-1}}}$$

ahol az N_x az x n-gram előfordulásainak számát jelenti az adott kategóriában.

Gondot okozhat, hogy nagyon sok: $O(W^n)$ paramétert kell becsülni, ahol W a szókincs mérete. Ennek a problémának a kezelésére csak azt javasolják, hogy vegyünk kis n-eket. Ez a legfőbb baj az n-gram modellel [17].

Ez alapján a *Markov*-lánccal bővített ML-osztályozó a következő:

$$\begin{aligned} c^* &= \arg\max_{c\in C}\{P(d|c)\} \\ &= \arg\max_{c\in C}\{P(w_1\dots w_T|c)\} \\ &= \arg\max_{c\in C}\{\prod_{i=1}^{T} P(w_i|w_{i-n+1}\dots w_{i-1}|c)\} \end{aligned}$$

Ezzel a képlettel azt a kategóriát kapjuk meg, amellyel legvalószínűbben generálhattuk az adott szöveget.

Az n-gram modellek kapcsolhatók az ún. *frázissúlyozáshoz*. Az n-gramokat tekinthetjük frázisoknak, a súlyukat pedig simított gyakoriság adja meg.

4.3 Szavak és karakterek

Az, hogy a szövegosztályozás szavakra épül, felveti a *tokenizálás*nak, azaz a szöveg szavakra bontásának problémáját.

Legegyszerűbb esetben szóna a szóközzel határolt nemszóköz-sorozatokat veszik [29].

Megfontolandó, hogy nem érdemesebb-e alapegységnek tekinteni a karaktert. Ezáltal a tokenizálás problémája kiküszöbölhető – ami bizonyos nyelveknél, például a kínainál meglehetősen nehéz –, ez a megközelítés csak annyit feltételez a szövegről, hogy bájtsorozatként leírható, semmilyen nyelvfüggő előfeldolgozásra nincs szükség [7].

Ha a karaktereket tekintjük alapegységnek, a modell ugyanaz, csak a szókincs sokkal kisebb, az n-gramok hosszát viszont érdemes lehet nagyobbra venni, ami megintcsak túl sok paraméterhez vezethet [29].

4.4 Mérőszámok

A szövegosztályozás teljesítményének mérésére többféle mérőszámot dolgoztak ki.

Nézzük a speciálisan a spamszűrés esetére felírt definíciókat. Jelöljük N_A-val az A kategóriájú levelek számát, $n_{A\to B}$-vel pedig azoknak a leveleknek a számát, melyek A kategóriájúak és az osztályozó B kategóriába osztotta be őket [1], így $N_A = \sum_B n_{A\to B}$. Jelöljük a spamek kategóriáját S-sel, a rendes levelek kategóriáját R-lel.

A dokumentumok hány százalékát osztályoztuk helyesen:

$$\textit{helyesség (accuracy)} = \frac{n_{R\to R} + n_{S\to S}}{N_R + N_S}$$

A megjelölt spamek hány százaléka volt ténylegesen spam:

$$\textit{pontosság (precision)} = \frac{n_{S\to S}}{n_{S\to S} + n_{R\to S}}$$

A tényleges spamek hány százalékát jelöltük meg helyesen:

$$\textit{lefedettség (recall)} = \frac{n_{S\to S}}{n_{S\to S} + n_{S\to R}}$$

A klasszikus szövegosztályozás esetén a szimmetrikus mérőszámok megfelelnek, spamszűrés esetén azonban mindenképpen részrehajlónak kell lennünk: a fals pozitív ($R \to S$) sokkal nagyobb hibát jelent, mint a fals negatív ($S \to R$). Megfelelő teljesítménymérő lehet, ha elsősorban a pontosságot, másodsorban pedig a lefedettséget vesszük figyelembe.

Az [1] cikkben a fals pozitívok csökkentésének érdekében kicsit megváltoztatták az eredeti MAP-NBC-t. Az eredeti

$$P(C = \text{spam}|d) > P(C = \text{rendes}|d)$$

helyett a

$$\frac{P(C = \text{spam}|d)}{P(C = \text{rendes}|d)} > \lambda$$

feltétel teljesülése esetén ítélik a levelet spamnek. A λ itt azt jelenti, hogy hányszor rosszabb elveszteni egy rendes levelet, mint kapni egy spamet. Az eredeti feltétel természetesen a $\lambda = 1$ esettel azonos, $\lambda > 1$ használatával részesíthetjük előnyben a rendes levelek kategóriáját.

Ennek segítségével speciálisan a spamszűrés értékelésére vezették be a TCR (Total Cost Ratio) mérőszámot.

$$\mathrm{TCR} = \frac{N_S}{\lambda \cdot n_{R \to S} + n_{S \to R}}$$

A TCR = 1 azt jelenti, hogy ugyanott vagyunk, mintha nem használnánk szűrőt. TCR > 1 esetén működik hasznosan a szűrő, annál hasznosabban, minél nagyobb az érték. Nehézséget jelenthet a λ értékének meghatározása, illetve a különböző egynél nagyobb TCR értékek összehasonlítása, interpretálása.

4.5 Eredmények

Ebben a fejezetben a szövegosztályozó programok teljesítményét és a működésükkel kapcsolatos tapasztalatokat tekintem át. Minden esetben azokat az eredményeket közlöm, melyek eléréséhez a lehető legkevesebb nyelvfüggő, vagy egyéb bedrótozott elemet használtak. Először tekintsük az általános szövegosztályozási feladatra felkészített alkalmazásokat.

A nyelv felismerése volt a cél a [7] cikkben, azaz az egyes osztályokba különféle nyelvű szövegek tartoztak. Tapasztalat, hogy a nyelv azonosítására készített program sok esetben a szöveg témáját észleli. Mindössze 50 kbyte tanítószöveg elég volt a megfelelő szintű (helyesség = 0,92) eredmény eléréséhez. Nagyon valószínűtlen, hogy ilyen kevés adat alapján jól lehet becsülni a paramétereket, az is igaz, hogy a döntésben fontos szerepet játszó paraméterek sokkal hamarabb konvergáltak, mint a kevésbé fontosak. Megállapítható, hogy egyéb korpuszalapú munkákkal ellentétben itt meglehetősen kicsi korpusz elégséges.

A [29] cikkben többek között görög szerzők műveit osztályozták, egy osztály itt egy szerző műveit jelentette. Arra az eredményre jutottak, hogy a sima NBC adta a legjobb eredményt (helyesség = 0,96), a markovi kiterjesztés ezen az eredményen nem javított. A karakteralapú markovi kísérletben harmadrendig (n = 3-ig) nőtt a helyesség, maximális értéke 0,9 volt. A megtorpanást azzal magyarázzák, hogy $n = 4$ esetén már túl ritkák az adatok ahhoz, hogy az $O(W^4)$ darab paraméter becslése kielégítő legyen. Megállapítják, hogy ha nincs elég tanítóadat, fontos csökkenteni az n értékét, hogy elkerüljük ezt a problémát. A modellhez mindig meg kell találni az optimális rendet [7]. Ezzel magyarázható, hogy miért részesítik előnyben az NBC-t, azaz az elsőrendű esetet a gyakorlatban.

Most nézzük a speciálisan spamszűrésre tervezett programokat[1].

[1] A továbbiakban a teljesítmény bemutatására az itt látható formát használom: először a levelek darabszámát közlöm, majd az osztályozás eredményét lehetőség szerint darabszámokkal, pontossággal és lefedettséggel (vö. 4.4. fejezet)

A [38] cikkben laboratóriumi körülmények között a következő eredményeket kapták:

	spam	rendes
tanító	1538	
tesztelő	251	
összesen	211	1578

pontosság = **0,971**
lefedettség = 0,943

Kipróbálták a programot valós körülmények között egy felhasználó leveleivel, ezzel az eredménnyel:

	spam	rendes
tanító	2593	
tesztelő	45	177
összesen	2815	

$n_{S\to S}$ = 36
$n_{R\to R}$ = 174
$n_{S\to R}$ = 9
$n_{R\to S}$ = 3

pontosság = **0,923**
lefedettség = 0,800

Megvizsgálták a három fals pozitív levelet: egy metaspam (ld. 6.4.1. fejezet) volt, kettő pedig valamilyen kevésbé fontos hírlevélből származó üzenet. Megállapították, hogy ezen levelek elveszése nem járt volna fontos információ elvesztésével. Úgy gondolták, hogy ezek az eredmények alkalmassá teszik az algoritmust kereskedelmi levelezőprogramokban való használatra is.

Az [1] cikk szerint az imént idézett cikk eredményei ugyan lenyűgözőek, de azért ők jobb eredményekre jutottak:

	spam	rendes
tanító	2603	
tesztelő	290	
összesen	480	2413

pontosság = **0,995**
lefedettség = 0,769

A tanítógyűjtemény méretét 260-tól 2603-ig változtatva tízszeres keresztellenőrzést végeztek: a táblázatban szereplő mérőszámok átlagértékek.

A cikkben szereplő legjobb pontosságot, melyet a táblázatban feltüntettem $\lambda = 9$ esetén érték el (vö. 4.4. fejezet). Arra a következtetésre jutottak, hogy ha nem vállaljuk fel a spamek azonnali letörlésének kockázatát – ezt modellezték a $\lambda = 999$ esettel –, akkor a program nagy biztonsággal használható.

Végül lássuk a másodikgenerációs CRM114 spamszűrő program meglepően jó eredményeit [47]:

	spam	rendes
tanító	n/a	
tesztelő	1935	5849
összesen	n/a	

$n_{S\to S}$ = 1931
$n_{R\to R}$ = 5849
$n_{S\to R}$ = 4
$n_{R\to S}$ = 0

pontosság = **1,000**
lefedettség = 0,998

A programmal mindössze 500 kbyte tanítókorpusszal 99,9 százalékos helyességet lehetett elérni. A kiemelkedő teljesítmény mellett a program 20 kbyte/másodperc sebességgel dolgozza fel a leveleket 5 Mbyte memóriafelhasználás mellett. A szerző szerint ez még mindig túl lassú és túl drága egy profitorientált internetszolgáltató számára.

5 A *Graham*-féle spamszűrési módszer

Ebben a fejezetben *Paul Graham* szövegosztályozási technikákon alapuló (vö. 4. fejezet) másodikgenerációs (vö. 3.3.3. fejezet) spamszűrő módszerét mutatom be.

Graham megelégelve a spamek lélekölő olvasgatását és spamszűrő szabályainak állandó fejlesztgetését, egy olyan szűrő kifejlesztésébe fogott, amely automatikusan tanul. Fontosnak tartotta, hogy a probléma nem fogható fel egyszerű szövegosztályozásként, a fals pozitív találatok elkerülésére kell elsősorban törekedni, ennek érdekében a szűrőnek valamilyen szempontból aszimmetrikusnak kell lennie. Végül a levelekben található szavak spamvalószínűségeinek bayesi kombinációját használva, egy trükkösen módosított bayesi osztályozóval kiemelkedően jó teljesítményt – 100 százalékos pontosság mellett 99,5 százalékos lefedettséget – ért el. A módszert 'A Plan for Spam' című dolgozatában tette közzé [12]. Ez a leírás azóta számos projektet ihletett, és az érdeklődés homlokterébe hozta a statisztikai spamszűrést. A bayesi módszer gyorsan a spamszűrés előszeretettel választott módszerévé vált [47].

5.1 Az algoritmus leírása

A módszer röviden leírható, de annál több fontos részletet rejt.

Két korpuszból indul ki, melyek a spam és a rendes leveleket tartalmazzák a fejlécsorokkal együtt.

A szöveget szavakra bontja úgy, hogy csak a betűket, számokat, kötőjelet, aposztrófot és a dollár jelet tekinti szó részének, a többi karakter mind szóelválasztónak minősül. A csak számokból álló szavakat elhagyja, és kiszűri a HTML-megjegyzéseket.

Megszámolja a szavak előfordulását a két korpuszban, és eltárolja külön két táblázatban. Egy harmadik táblázatban kiszámolja a gyakoribb szavakhoz tartozó spamvalószínűséget – azaz azt, hogy az adott szót tartalmazó levél milyen eséllyel spam – a következő módon:

Jelölje az x_i szó esetében g a rendes levelek közötti előfordulási szám *dupláját*, b a spamlevelek közötti előfordulási számot, N_R a rendes levelek, N_S pedig a spamek számát. Ha $g + b > 5$, akkor bekerül a táblázatba az x_i szó a

$$P(C = \text{spam}|X_i = x_i) = \max\left(0{,}01; \min\left(0{,}99; \frac{\min\left(1, \frac{b}{N_S}\right)}{\min\left(1, \frac{g}{N_R}\right) + \min\left(1, \frac{b}{N_S}\right)}\right)\right)$$

értékkel (vö. 5.2.1. fejezet). Ez a táblázat képezi a nyelvi modellt, ami alapján a döntéseket hozzuk. Beérkező levél esetén kiválasztja a levélből azt

a 15 szót, melyeknek a spamvalószínűsége legtávolabb van a semleges $\frac{1}{2}$-től. Ebből a 15 értékből számítja a teljes levél spamvalószínűségét a *Bayes*-tétel szerint (vö. 5.2.2. fejezet):

$$P(C = \text{spam}|X = x) = \frac{\prod_i P(C = \text{spam}|X_i = x_i)}{\prod_i P(C = \text{spam}|X_i = x_i) + \prod_i (1 - P(C = \text{spam}|X_i = x_i))}$$

Ha ez az érték a 0,9-es küszöböt meghaladja, akkor a levelet spamnek ítéli, egyébként elfogadja rendes levélnek [12].

5.2 A módszer összevetése az NBC-vel

Az algoritmus egyszerűsége és intuitivitása mellett az NBC-k kapcsán felmerülő számos technikát alkalmazza, igaz, sokszor egyszerűsített formában (vö. 4. fejezet). Bizonyos helyeken eltér az eredeti NBC-től, ezt szinte mindig az algoritmus előre tervezett, a fals pozitív találatok elkerülése érdekében megvalósított aszimmetrikusságával indokolja a szerző (vö. 5. fejezet). Az elemzés során többször reagálok a [36] cikk *Graham*-módszert kritizáló megjegyzéseire.

5.2.1 Algoritmus és modell

Az eredeti NBC-eljáráshoz képest fontos különbség, hogy ML-osztályozó helyett küszöbölési technikát alkalmaz. Ezenkívül a g értékében megjelenő drasztikus kétszeres szorzó talán az egyetlen technika, amivel az NBC-k között nem találkoztam. Felmerülhet, hogy miért éppen kettővel kell megszorozni a rendes levelek szavainak az előfordulási számát a jó döntéshez. Ki lehet próbálni más értékeket, lehetőség nyílik az algoritmus fejlesztésére.

Az osztályozó tanítására itt is felügyelt tanulásos keretrendszer szolgál.

Szövegmodell tekintetében nagyjából a polinomiális eseménymodell (ld. 25. o.) szerint dolgozik, azaz tekintetbe veszi a szavak előfordulási számát is, de csak addig a pontig, míg a levelek számát nem lépi túl, azaz a paramétereket az eredeti

$$P(X_i = x_i|C = \text{spam}) = \frac{n_i^S}{\text{a spamek össz-szószáma}}$$

formula helyett a

$$P(X_i = x_i|C = \text{spam}) = \min\left(1, \frac{n_i^S}{N_S}\right)$$

módon becsüli. n_i^S itt az x_i előfordulásainak számát jelenti a spamek között.

Graham algorimusában az eredeti NBC-hez képest végig a fordított irányú feltételes valószínűségek szereplenek, ezek egyszerűen adódnak *Bayes*-tétellel, egyenletes prior eloszlás – $P(C = \text{spam}) = \frac{1}{2}$ – feltételezése mellett.

$$\begin{aligned} P(C = \text{spam}|X_i = x_i) &= \frac{P(X_i = x_i|C = \text{spam}) \cdot P(C = \text{spam})}{P(X_i = x_i)} \\ &= \frac{P(X_i = x_i|C = \text{spam})}{2 \cdot P(X_i = x_i|C = \text{rendes}) + 2 \cdot P(X_i = x_i|C = \text{spam})} \\ &= \frac{\min\left(1, \frac{n_i^S}{N_S}\right)}{2 \cdot \min\left(1, \frac{n_i^R}{N_R}\right) + 2 \cdot \min\left(1, \frac{n_i^S}{N_S}\right)} \end{aligned}$$

Mivel $b = n_i^S$ és $g = 2 \cdot n_i^R$, közelítőleg megkaptuk a spamvalószínűségek kiszámítására használt

$$P(C = \text{spam}|X_i = x_i) = \frac{\min\left(1, \frac{b}{N_S}\right)}{\min\left(1, \frac{g}{N_R}\right) + \min\left(1, \frac{b}{N_S}\right)}$$

képletet (vö. 5.1. fejezet).

5.2.2 Bayesi egyáltalán a módszer?

A [36] cikk szerint az algoritmus sehogy sem tekinthető bayesinek. Egy levél spamvalószínűségét megadó formula levezetésével bizonyítom be az állítás ellenkezőjét. Az 5.1. fejezet végén található egyenlőséget kell igazolni:

$$P(C = \text{spam}|X = x)$$

ez a *Bayes*-tétel szerint,

$$\frac{P(X = x|C = \text{spam}) \cdot P(C = \text{spam})}{P(X = x)}$$

a teljes valószínűség tétele szerint tovább alakítva,

$$\frac{P(X = x|C = \text{spam}) \cdot P(C = \text{spam})}{\sum_{c \in C} P(X = x|C = c) \cdot P(C = c)}$$

feltesszük, hogy az a priori eloszlás egyenletes, azaz $P(C = \text{rendes}) = P(C = \text{spam}) = \frac{1}{2}$

$$\frac{P(X = x|C = \text{spam})}{\sum_{c \in C} P(X = x|C = c)}$$

a szummát felbontva,

$$\frac{P(X = x|C = \text{spam})}{P(X = x|C = \text{spam}) + P(X = x|C = \text{rendes})}$$

ismét a *Bayes*-tétel alkalmazásával,

$$\frac{\frac{P(C=\text{spam}|X=x)\cdot P(X=x)}{P(C=\text{spam})}}{\frac{P(C=\text{spam}|X=x)\cdot P(X=x)}{P(C=\text{spam})} + \frac{P(C=\text{rendes}|X=x)\cdot P(X=x)}{P(C=\text{rendes})}}$$

az a priori eloszlás egyenletessége miatt $P(C = c)$-k kiesnek, és $P(X = x)$-szel is lehet egyszerűsíteni,

$$\frac{P(C = \text{spam}|X = x)}{P(C = \text{spam}|X = x) + P(C = \text{rendes}|X = x)}$$

most feltesszük függetlenséget,

$$\frac{\prod_i P(C = \text{spam}|X_i = x_i)}{\prod_i P(C = \text{spam}|X_i = x_i) + \prod_i P(C = \text{rendes}|X_i = x_i)}$$

végül mivel $1 - P(C = \text{spam}|X_i = x_i) = P(C = \text{rendes}|X_i = x_i)$

$$\frac{\prod_i P(C = \text{spam}|X_i = x_i)}{\prod_i P(C = \text{spam}|X_i = x_i) + \prod_i (1 - P(C = \text{spam}|X_i = x_i))}$$

így megkaptuk a kívánt formulát.

Azért kell a *Bayes*-tételt kétszer alkalmazni, mert *Graham* megszövegezése a klasszikus $P(x|c)$ valószínűségek helyett a $P(c|x)$-ekkel dolgozik [30].

5.2.3 Függetlenségi feltételezés

Az algoritmusban függetlennek tekintettük egymástól az egyes szavak előfordulását. A [36] cikk ezt is jogtalannak tartja és sérelmezi, pedig olyan gyakran alkalmazott fogás, hogy külön elnevezése is van, emiatt hívják „naív" bayesi osztályozónak az NBC-t.

Régi hagyománya van a függetlenségi feltételezés megsértésének, ennek legfőbb oka, hogy a módszer ezzel együtt nagyon jól működik a gyakorlatban [13] (vö. 4.1.1. fejezet).

5.2.4 Prior eloszlás

A [30] írás amiatt támadja *Graham*et, hogy a spamek és a rendes levelek valószínűségét egyenlőnek veszi, azaz számításában egyenletes prior eloszlást feltételez. Pedig rendszerint, ha a prior eloszlás ismeretlen, akkor egyenletes eloszlást alkalmaznak. A függetlenségi feltételezéshez hasonlóan ez is annyira szokásos, hogy szintén van külön elnevezése. Az egyenletes prior feltevésével kapjuk a MAP-osztályozóból az ML-osztályozót [29] (vö. 24. o.).

Nem állják meg a helyüket azok a „vádak" [30], hogy a szerző ezzel eltorzította az NBC eredményét vagy hogy figyelmetlenségből hagyta ki a számításból a P(rendes), P(spam) értékeket. *Graham* elismeri [10], hogy a számítást ez megkönnyítette (vö. 5.2.2. fejezet), emellett hangsúlyozza, hogy azért vette elő az egyenletes prior eloszlást, mert a P(rendes) és P(spam) értékek becslése egyáltalán nem triviális. Az általánosságban számított valószínűség félrevezetne, ugyanis a spamek mennyisége a nap folyamán is nagymértékben változó. A becsléseket legalább óránkénti bontásban kellene megadni [13].

5.2.5 Tisztítás

Tisztításkor elhagyjuk a túl ritkán előforduló szavakat, így kerüljük el a pontatlanul becsült paraméterek használatát.

Ezzel a módszerrel – esetleg kis teljesítménycsökkenés árán – a nyelvi modell mérete drasztikusan csökkenthető [19].

A *Graham*-féle módszerben a tisztítást a $g + b > 5$ feltétel valósítja meg, ti. az, hogy csak az együttesen 5-nél többször előforduló szavak kerülnek bele a nyelvi modellbe.

5.2.6 Tulajdonságkiválasztás

Szintén gyakran alkalmazott eljárás, melynek során egy nagyon sok tulajdonsággal rendelkező objektum tulajdonságai közül megpróbáljuk kiválasztani azokat, amelyek alapján jó eséllyel helyes döntést hozhatunk. Ezeket nevezhetjük lényeges tulajdonságoknak.

Modellünkben az emailek „tulajdonságai" a szóelőfordulások. *Graham* a levelekből azt a 15 szót választja ki, amelyeknek a spamvalószínűsége leginkább eltér a semleges $\frac{1}{2}$-től, és csak ezek alapján – a legártatlanabb és a leggyanúsabb szavak alapján – dönt.

A [36] felveti, hogy az osztályozásnál a levél összes szavát figyelembe kellene venni. Ez azért nem jó, mert könnyen átjuttathatja a spamelő a levelét a szűrőn, ha nagy terjedelmű véletlen szöveggel egészíti ki. Ugyanis a számításban elsikkad a néhány spamgyanús szó, a hatalmas tömegű semleges szó mellett. Éppen a tulajdonságkiválasztás miatt marad az az egyetlen lehetőség, hogy csak ártatlan – rendes levelekre jellemző – szavakkal lehet ellensúlyozni a spamgyanús szavakat. Ezeket viszont a spamelő természetesen nem ismeri, mivel nincs beletekintése a felhasználók személyes levelezésébe [15].

Graham lényegében nem indokolja meg, hogy miért pont 15 szót választ ki. Túl keveset választani nem jó, mert minden levél tartalmaz néhány gyanús szót, túl sokat választani sem jó, mert a spamek is tartalmaznak semleges

szavakat. [13] Itt lehetőség nyílhat az algoritmus fejlesztésre.

Véleményem szerint ez a fajta tulajdonságkiválasztás a módszer egyik kulcsa, a kimagasló teljesítmény alapja lehet.

5.2.7 Simítás

Az NBC esetén valahogy biztosítani kell, hogy a nyelvi modellben nem szereplő, illetve valamelyik kategóriában nem szereplő szavak esetén ne legyen a paraméter becslése nulla, mert ez rögtön a kategória elvetésével jár.

Mivel *Graham* nem ML-osztályozót használ, erre itt nem lenne szükség, mégis az összes paraméterbecslést a 0,01 és a 0,99 értékek közé szorítja be.

A [36] cikk – melynek egyébként számos kritikai észrevételét megalapozatlannak találtam – megjegyzi, hogy nem sok értelme van a nyelvi modellben egyáltalán nem szereplő szavaknak 0,4-es spamvalószínűséget adni, hiszen az ötnél kevesebbszer előforduló szavakat egyáltalán nem vesszük tekintetbe. Ebben teljesen igaza van.

5.3 Eredmények

Nézzük, hogyan teljesített ez a módszer a valós kipróbálás során. *Graham* saját levelein végzett tesztjének eredménye [10]:

	spam	rendes
tanító	4000	4000
tesztelő	1750	n/a
összesen	5750	n/a

$n_{S \to S} = 1746$
$n_{R \to R} = \text{n/a}$
$n_{S \to R} = 4$
$n_{R \to S} = 3$

pontosság = **0,998**
lefedettség = 0,998

Az eredmény tényleg nagyon jó. A három fals pozitívról a szerző megállapítja, hogy kettő olyan volt, amit egyáltalán nem bánt volna, ha elvész, a harmadik viszont egy csupa nagybetűvel írt, Egyiptomból feladott levél volt, ami fontos információkat tartalmazott.

A bayesi szövegosztályozás régi, elfogadott módszer. Az az újdonság, hogy ez az algoritmus a spamszűrésben ilyen hatékony tud lenni [13]. A bayesi spamszűrők talán azért nem terjedtek el eddig, mert a korábbi próbálkozások [27, 38] nem biztosítottak elfogadható teljesítményt. Ennek oka lehet, hogy túl kicsi korpusszal tanítottak, elhagyták a fejlécsorokat, a szavakat szótővé alakították, nem alkalmaztak tulajdonságkiválasztást, és nem tettek kellő erőfeszítéseket a fals pozitív találatok elkerülésére [10].

A SpamBayes projektben kifejezetten a most ismertetett módszert alkalmazzák, és nem a klasszikus NBC-t. Befejezésképpen lássuk egy fejlesztő véleményét a módszerről.

„Sose írtam még egy ilyen zűrös, valódi életbeli problémára ilyen kis erőbefektetéssel ilyen jól működő programot. Szövegosztályozásra talán nem lenne alkalmas a *Graham*-féle módszer, de spamszűrésre hihetetlenül jó. Szerintem ez egy zseniális ötlet." – lelkendezik *Tim Peters* a SpamBayes fejlesztője [31]

6 A módszer kiterjesztése – az SHF szűrő

A tervem az volt, hogy a dolgozat keretében egy grahami elv szerint működő spamszűrőt készítek, az eredetit annyiban kiterjesztve, hogy egyes szavak gyakorisága helyett szópárok, szó-n-esek – n-gramok – gyakoriságára építsen. Ennek a változatnak a teljesítményét összehasonlítom az eredetiével.

Ezenkívül az algoritmusban elrejtett „mágikus" számokat paraméternek tekintve, megpróbálom kikísérletezni a legjobb teljesítményt adó paraméterkombinációt. Itt elsősorban a rendes szavak előfordulását érintő kétszeres szorzóra (ld. 5.2.1. fejezet), illetve a tulajdonságkiválasztásnál használt 15-ös darabszámra (ld. 5.2.6. fejezet) gondoltam.

Az n-gramokra való kiterjesztés ötletét *Graham* veti fel, bízva abban, hogy a kiterjesztés a valószínűségek pontosabb becslését, így jobb teljesítményt eredményez. „Egy szópárokon alapuló szűrő végeredményben éppen egy *Markov*-láncos szöveggeneráló program megfordítása lenne." – állapítja meg [12]. Ez a módszer hasonlít a frázissúlyozáshoz: az n-gramok tekinthetők frázisoknak, súlyukat pedig a simított gyakorisággal lehet becsülni [29].

Nyelvfüggő vagy egyéb bedrótozott elemeket a legmesszebbmenőkig próbáltam elkerülni, mert az ilyesmi mindig csökkenti a program alkalmazkodóképességét. A *Graham* által használt tokenizálót egyszerűsítettem, a csak számból álló szavakat nem hagytam el, a HTML-kódot, sőt a csatolt fájlokat is meghagytam, a HTML-megjegyzéseket sem töröltem, semmit nem változtattam a leveleken.

6.1 A program működése

Két korpusz – egy spam- és egy rendes levél korpusz – segítségével betanítjuk a programot, felépítjük a nyelvi modellt (az adatbázist) (`SHFlearn.pl`). Ez alapján döntünk a beérkező levelekről (`SHFapply.pl`), megállapítjuk, hogy a levél spam-e vagy nem. Mikor összegyűlt egy újabb adag levél, újbóli tanulással bővíteni lehet az adatbázist.

6.2 Felhasználói útmutató

Dolgozatom lényege nem a kényelmes felhasználói verzió kifejlesztése, hanem a *Graham*-féle spamszűrési módszer vizsgálata volt, emellett akinek van kedve hozzá, nyugodtan kipróbálhatja a programot. A programot tartalmazó `SHF.zip` fájl a `http://shf.azbest.hu` címről tölthető le, és a mellékelt lemezen is megtalálható. Ez a kísérletek során kialakult legjobb paraméterkombinációt tartalmazza. A program *Linux* operációs rendszer felett fut, a *Pine* levelezőprogrammal való együttműködésre van felkészítve. Szükségesek

hozzá a *Procmail* és a *Formail* programok.

6.2.1 Installálás

Az `SHF.zip` fájlt kicsomagoljuk, és a benne található `README` fájl szerint járunk el: létrehozzuk a `~/.SHF` könyvtárat, hozzáadjuk a `PATH`-hoz, belemásoljuk a `.pl` és `.pm` fájlokat, a `~/.procmailrc` fájlunkat pedig kiegészítjük a `.procmailrc.SHF` fájllal (ld. az A.5. függelékben). Ha nincs `~/.procmailrc` fájlunk, akkor egyszerűen hozzuk létre a `.procmailrc.SHF`-fel azonos tartalommal.

6.2.2 Tanítás

Az `SHFlearn.pl` tanító scriptnek első paraméterként a megtanulandó levélgyűjteményfájlt adjuk meg teljes elérési úttal, második paraméterként `OK`-t vagy `SPAM`-et írunk, aszerint, hogy a megadott gyűjtemény milyen kategóriájú leveleket tartalmaz. Ha már van kész nyelvi modell, akkor kiegészíti, ha nincs, létrehozza. A teljes nyelvi modellt a `DB_full` fájlban tárolja, külön tárolja a gyakoribb k-gramokat a `DB` fájlban, ezt használja szűréskor. A tanítás mindig a `DB_full` fájlból indul ki, és ha valamelyik szó épp átlépi a minimális előfordulási küszöböt, akkor belekerül a `DB`-be. Ha a teljes adatbázist helyhiány miatt letöröljük, akkor a tanítás a `DB` fájlt fogja kiegészíteni.

6.2.3 Alkalmazás

Ha megfelelően installáltuk és betanítottuk a programot, akkor automatikusan szűrni kezdi a beérkező leveleket: a szűrő ítélete meg fog jelenni az `X-SpamHammerFiltered:` fejlécsorban, ami alapján könnyen külön mappába irányíthatjuk a spamleveleket (vö. A.5. függelék), illetve a tárgysorban is `[SHF:OK]` vagy `[SHF:SPAM]` formában. Ha nincs kész nyelvi modell, akkor nem működik a szűrés, és a program hibajelzést sem ad.

6.3 Kísérletek

Ebben a fázisban a cél az algoritmusban lévő „mágikus” számok módosítása volt, a nagyobb teljesítmény reményében.

Ezeket a számokat mind paraméternek vettem, ezekhez jön hozzá még egy paraméter: a markovi kiterjesztés rendje, a k, ami a k-gramok hosszát jelöli.

A továbbiakban a $k = 1$ esetre bayesi, a $k > 1$ esetre markovi esetként hivatkozom. A markovi esetben *Graham* javaslata szerint az egész algoritmus

azonos az eredetivel, csak nem tokenek, hanem k-gramok gyakoriságait tartalmazza a nyelvi modell. A cél éppen az volt, hogy megvizsgáljuk, hogy a várakozásainknak megfelelően a markovi esetben tényleg jobb teljesítményt lehet-e elérni.

A következő hat paraméter javításával próbálkoztam:

- `MODE`: az alapegység: szó (WORDS) vagy karakter (CHARS);
- `k`: a k-gramok hossza;
- `OK_FACTOR`: hányszoros szorzóval szerepeljenek a rendes levelek szavainak gyakorisági értékei a nyelvi modellben;
- `MIN_APPEAR`: tisztítás: egy k-gram minimum hányszor forduljon elő a nyelvi modellben ahhoz, hogy szerepelhessen a kiértékelésben;
- `fac`: a tulajdonságkiválasztás után hány darab k-gram kerüljön a valószínűségi számításba;
- `SPAM_THRESHOLD`: milyen valószínűségi határ fölött tekintjük spamnek a levelet.

A kísérletek eredményét mindig a *pontosság* – a spamnek jelölt levelek hány %-a ténylegesen az – és a *lefedettség* – a tényleges spamek hány %-át találta meg – mérőszámokkal értékeltem (ld. 4.4. fejezet). A spamszűrésben a teljesítményt elsősorban a pontosság, és csak másodsorban a lefedettség jellemzi. Úgy tekintettem, hogy ha a pontosságon egy kicsit is javítani tudunk (azaz még kevesebb rendes levelet nyilvánítunk spamnek), akkor a lefedettség romlása (néhány spamet rendesként átengedünk) kevésbé fontos. Ezzel biztosítottam a spamszűrőktől megkövetelt alapvető tulajdonságot, hogy rendes leveleket lehetőleg soha ne tekintsen spamnek.

Kétféle tesztelést alkalmaztam:

Körbetesztelés: kiválasztottam 100 db spam és 100 db rendes levelet. Kivettem egy levelet, a többi 199-cel betanítottam a szűrőt, és megnéztem, hogy az egy kiválasztott levelet hogyan ítéli meg. Így végigmentem mind a 200 levélen. Ez felfogható 200 különböző nyelvi modellel való tesztelésnek. A módszer viszonylag hosszú időt vett igénybe, mivel minden egyes tesztelési lépés előtt újra kell tanítani a szűrőt.

Fix tesztelés: véletlenszerűen ketté bontottam a leveleimet, a levelek $\frac{4}{5}$-e lett a tanító és $\frac{1}{5}$-e a tesztelő gyűjtemény. Tanítóval betanítottam, tesztelővel leteszteltem, a véletlenszerű vágást és az egész eljárást 25-ször végeztem el, és eredményként a mérőszámok átlagát jelenítettem meg.

Körbeteszteléssel állapítottam meg, hogy mely paraméterkombináció adja a legjobb teljesítményt, majd ezt a kombinációt fix teszteléssel is ellenőriztem.

A felhasznált levélgyűjtemény a saját levelezésem anyaga: 865 rendes és 197 spamlevél.

A SpamBayes [16] projektben is használt próbaalapú fejlesztést alkalmaztam, változtattam a paraméterértékeket és azt vizsgáltam, hogy ez milyen hatással van a teljesítményre. A paraméterek következő értékeire futtattam le a tesztet. Az eredeti *Graham*-féle értékeket **vastagítva** emeltem ki.

név	*érték*
MODE	**WORDS**; CHARS
k	**1**; 2; 3; 4; 5; 6
fac	10; **15**; 20
OK_FACTOR	1; 1,25; 1,5; **2**
MIN_APPEAR	**5**; 8
SPAM_THRESHOLD	0,8; **0,9**

6.3.1 Eredmények

Körbeteszteléssel a következő eredményeket kaptam:

Szóalapú megközelítésnél (MODE = WORDS), a bayesi – $k = 1$ – esetben a legjobb eredmény

$$\begin{array}{lcr} \textbf{pontosság} & = & \textbf{1,000} \\ \text{lefedettség} & = & 0{,}950 \end{array}$$

lett,

fac	20◀
OK_FACTOR	1,5◀
MIN_APPEAR	5
SPAM_THRESHOLD	mindegy

paraméterértékekkel. Az eredetitől eltérő értékeket◀ jelöli.

A markovi esetben $k = 2$-re kaptam a legjobb eredményt

$$\begin{array}{lcr} \textbf{pontosság} & = & \textbf{1,000} \\ \text{lefedettség} & = & 0{,}930 \end{array}$$

A paraméterértékek a következők voltak:

fac	15
OK_FACTOR	1,5◀
MIN_APPEAR	8◀
SPAM_THRESHOLD	0,9

A karakteralapú megközelítésnél (`MODE` = CHARS), `OK_FACTOR` = 1,5-tel és egyébként az eredeti paraméterek mellett a következő eredményeket kaptam:

`k`	pontosság	lefedettség
2	0,705	0,980
3	0,907	0,970
4	0,951	0,970
5	0,933	0,980
6	0,950	0,960

Mivel a karakteralapú esetben kapott pontosságértékek a spamszűrésnél nem elfogadhatók, ezért fix teszteléssel csak a fenti két szóalapú eredményt és az eredeti grahami paraméterkombinációt ellenőriztem:

kísérlet	pontosság	lefedettség
eredeti *Graham*	0,994	0,858
$k = 1$ legjobb	0,999	0,894
$k > 1$ legjobb	0,982	0,869

6.3.2 Elemzés

Általánosságban látszik, hogy a `SPAM_THRESHOLD`, azaz a küszöbparaméter kis megváltoztatásának nincs hatása a teljesítményre . *Graham* szerint nagyjából mindegy, hogy hol a küszöb, mert a valószínűségek általában 0 vagy 1 közeliek [15]. Ezt az én tapasztalataim is igazolják: a rendes levelek spamvalószínűsége általában kisebb, mint 10^{-40}, a spameké pedig nagyobb, mint $1 - 10^{10}$.

A másik három paraméter – `fac`, `OK_FACTOR` és `MIN_APPEAR` – viszont fontos szerepet játszott. A `MODE` és `k` értékek változtatásával más-más kombinációk bizonyultak jónak.

Egy kurrens cikk szerint az, hogy a teljesítménynövekedés egy bizonyos k értéknél megáll, azzal magyarázható, hogy nagy k-ra már túl ritkák az adatok, így az $O(W^k)$ paraméter becslése sokkal kevésbé lehet pontos [29]. Kisméretű nyelvi modellhez kis k értéket válasszunk; általánosságban is elmondható, hogy mindig van egy optimális k a modellhez [7].

A karakteralapú esetben jó lefedettségértékek mellett $k = 4$-ig nőtt a pontosság, azonban a kapott legjobb pontosságérték (0,951) a spamszűrésben nem elfogadható. Nem engedhető meg, hogy a spamek között minden huszadik egy tévesen osztályozott rendes levél legyen. Így a karakteralapú megközelítést a továbbiakban elvetettem.

A szóalapú esetben a markovi kiterjesztés nem hozott teljesítményjavulást, leghatékonyabban a bayesi osztályozó működött, azaz az optimális k ebben az esetben a $k = 1$. Ez magyarázatként szolgálhat arra, hogy miért részesítik előnyben a bayesi szövegosztályozót a gyakorlatban.

A markovi kiterjesztés kapcsán kapott eredményeim teljesen összecsengenek a [29] cikk eredményeivel (vö. 30. o.).

Dolgozatom egyik eredménye, hogy a bayesi esetben az eredeti *Graham*-féle paraméterkombinációnál jobbat – `fac` = 20 és `OK_FACTOR` = 1,5 – találtam. *Graham* nem nagyon indokolja, hogy ezen paraméterek esetében miért éppen az általa választott értékek mellett döntött, inkább csak a megérzéseire hagyatkozik. Én a paraméterválasztásomat a fenti empirikus teljesítményvizsgálatok eredményeivel tudom alátámasztani és indokolni. A továbbiakban és a végleges programváltozatban is ezen paramétereket alkalmaztam.

6.3.3 Az ember spamfelismerő képessége

A kísérletek során kiderült, hogy a spamkorpuszba véletlenül belekerült egy rendes levél is. Annak idején elolvasás nélkül, csak spamgyanús tárgya alapján ítélve tettem tévesen a spammappába. Nagyon fontos eredménynek tartom, hogy a programom ezt a levelet bármely paraméterbeállítás mellett rendes levélként határozta meg. Általános felfogás, hogy a címzett mindig könnyen felismeri a spamet [12]. Általában etalonnak – kimondatlanul 100%-os pontosságúnak és lefedettségűnek – tekintik az ember spamfelismerő képességét. Látjuk, hogy ez nincs egészen így, jelen esetben a programom nálam nagyobb pontossággal dolgozott. Az egyik cikk közlése szerint, az ember spamfelismerő képességének helyessége 99,84% [47]. A cikk velem ellentétben a helyességet (ld. 29. o.) használja a teljesítmény mérésére, így itt én is ezt adom meg. A spamfelismerésem helyessége:

$$\frac{n_{R\to R} + n_{S\to S}}{N_R + N_S} = \frac{865 + 196}{866 + 196} = 99{,}91\%$$

volt. Tanulság lehet, hogy egy mesterséges intelligencia-alkalmazásnak nem feltétlenül az emberi teljesítmény elérésére kell törekedni, néha ennél több is lehet a cél.

6.4 Tesztelés

Miután a kísérletek során az általam legjobbnak ítélt paraméterkombinációt meghatároztam, a programot valós működési körülmények között teszteltem.

A bayesi elven működő statisztikai spamszűrők tesztelésében problémát jelent, hogy körülményes hozzájutni mások személyes levelezéséhez. Általában

minden szerző csak a saját levelein teszteli a programját [13, 38]. Előfordul, hogy mesterségesen állítják össze a tesztelő készletet: az [1] cikkben például spamgyűjteményük mellé egy levelezőlista anyagából vették a rendes leveleket. Kétséges, hogy ez mennyire jól modellezi egy ember személyes levelezését, nem biztos, hogy az ilyen próbálkozásokból a valódi esetre vonatkozó következtetéseket lehet levonni. Ez kicsit hasonlít ahhoz, amikor valaki a szűrő tesztelése céljából megpróbál írni egy spamet; az a tapasztalat, hogy ez nem sikerül. Véleményem szerint csak úgy szabad tesztelni, hogy egy felhasználó személyes levelezésével tanítjuk a szűrőt, a tesztidőszakban pedig szigorúan minden beérkező levelét megvizsgáljuk.

Hárman voltak olyan szívesek és rendelkezésemre bocsátották teljes levelezésüket, így magamat is beleértve négy ember levelezésén állt módomban tesztelni a programot. A teszt során néhány héten keresztül beérkező leveleik közül nem töröltek le semmit, és a spameket külön gyűjtötték. Ezután két fájlban megkaptam a spam illetve a rendes leveleket. Adott időpontnál kettévágtam a fájlokat, az időpont előtti levelekkel tanítottam a szűrőt és az időpont utániakkal teszteltem.

Jelöljük a négy tesztesetet az S – mint saját –, A, B és C betűkkel.

S tesztesetben:

	spam	rendes
tanító	229	1557
tesztelő	13	157
összesen	242	1714

$n_{S\to S} = 13$
$n_{R\to R} = 155$
$n_{S\to R} = 0$
$n_{R\to S} = 2$

pontosság = 0,867
lefedettség = 1,000

A program minden spamet felismert, viszont két rendes levelet elutasított. Mindkét problémás levél metaspam (ld. 48. o.) volt. Kipróbáltam ugyanis a szabványos `abuse@domainnév` használatát [5], és a válaszlevelekben visszakaptam az összes spamet, amit küldtem nekik, ez csapta be a szűrőt.

A pontosság teljesen elfogadhatatlannak látszik, de ha nem követeljük meg a metaspam kezelését a szűrőtől, akkor 1-nek tekinthető. Kiemelendő, hogy 155 rendes levelet hiba nélkül átengedett a program.

A tesztesetben:

	spam	rendes
tanító	201	1955
tesztelő	42	119
összesen	243	2074

$n_{S\to S} = 41$
$n_{R\to R} = 119$
$n_{S\to R} = 1$
$n_{R\to S} = 0$

pontosság = 1,000
lefedettség = 0,976

A program egyetlen spam áltengedésétől eltekintve hibátlanul működött. Megjegyzendő, hogy itt is előfordult emberi osztályozási hiba, negyvenharmadikként egy rendes levél szerepelt a spamek között. Ezt a program kiszűrte,

azaz ismét bizonyos szempontból az emberi spamszűrő teljesítménynél jobban működött (vö. 6.3.3. fejezet) .

B tesztesetben:

	spam	rendes
tanító	1363	370
tesztelő	98	44
összesen	1461	414

$n_{S\to S} = 74$
$n_{R\to R} = 44$
$n_{S\to R} = 24$
$n_{R\to S} = 0$

pontosság = **1,000**
lefedettség = 0,755

A szűrő a spamek jelentős részét átengedte hibátlan pontosság mellett. A 24 átengedett levélből mindössze kettő volt klasszikus értelemben vett UBE (ld. 2.1.1. fejezet), ráadásul ez a kettő is a talán *subject-spam*nek nevezhető speciális csoportba tartozik: olyan levél, amelyeknek a törzse üres, a teljes üzenetet a tárgysor tartalmazza. A többi 22 levél munkatársaktól és levelezőlistákról kapott levelekből áll. Arra a kérdésemre, hogy valójában mi is volt itt a spam definíciója, a következő választ kaptam: „Lényegében azt vettem spamnek, amit pillanatnyi hangulatom szerint nem volt kedvem elolvasni. Az lenne jó, ha a program a kellemetlen leveleket megjelölné, és amiket szívesen olvasok, azokat nem." Ez meglehetősen szélsőséges definíciónak tekinthető, és ennek fényében az eredmény lehet, hogy nem is annyira lebecsülendő, a hibátlan pontosság mindenesetre figyelemreméltó. *Graham* egyik fontos előnyként emeli ki, hogy módszere lehetőséget nyújt mindenkinek, hogy a saját tetszése szerint definiálja a spamet. Elmondható, hogy nem szokásos spamdefiníció mellett is magasszintű pontossággal működik a program, csak több spamet enged át.

C tesztesetben:

	spam	rendes
tanító	500	298
tesztelő	172	40
összesen	672	338

$n_{S\to S} = 172$
$n_{R\to R} = 40$
$n_{S\to R} = 0$
$n_{R\to S} = 0$

pontosság = **1,000**
lefedettség = 1,000

Ez a vizsgálat nem tekinthető teljesértékűnek, mert nem kaptam meg az összes rendes levelet, csak két mappát. Úgy látszik, a spameket egy-két mappa egységes anyagától sokkal könnyebb elválasztani, mint a rendes levelek változatos összességétől.

Témavezetőm ötlete nyomán kipróbáltam a programot szövegosztályozóként is. *Graham* kategorikusan kijelenti, hogy a spamszűrés *nem* szövegosztályozás, és éppen az aszimmetrikussága miatt [10]. Mégpedig azért, mert egy spam átengedése sokkal kisebb baj, mint egy rendes levél kiszűrése. Itt az egyik korpusz egy baráti társaság leveleiből állt, a másik pedig egyéb rendes levelekből. A fentiek miatt a tesztet mindkét irányban elvégeztem, először

a baráti társaság leveleit vettem spamnek (D teszteset), majd fordítva (E teszteset).

D tesztesetben:

	spam	rendes
tanító	235	298
tesztelő	64	40
összesen	299	338

$n_{S\to S} = 64$
$n_{R\to R} = 37$ **pontosság = 0,955**
$n_{S\to R} = 0$ lefedettség = 1,000
$n_{R\to S} = 3$

A baráti társaság minden levelét helyesen ítélte meg, viszont három rendes levelet is a baráti társaság levelei – a „spamek" – közé sorolt. A hiba oka az lehet, hogy ezen három levél feladója egyben a baráti társaság tagja is.

Látszik, hogy a program spamszűrőként viselkedik: ha egy címről egyszer spam érkezik, akkor a többi onnan érkező levél is jogosan válik gyanússá, hiszen a klasszikus spamszűrésben szinte soha nem fordul elő, hogy egy adott emailcímről spam és rendes levél is érkezik (eltekintve a reverse spamtől).

E tesztesetben:

	spam	rendes
tanító	298	235
tesztelő	40	64
összesen	338	299

$n_{S\to S} = 31$
$n_{R\to R} = 64$ **pontosság = 1,000**
$n_{S\to R} = 9$ lefedettség = 0,775
$n_{R\to S} = 0$

Ebben az irányban is hasonló eredményt kaptam, *ugyanazt* a három levelet megint rosszul osztályozta, a különbség annyi, hogy egyéb leveleket is elfogadott a „rendes" kategóriába. A program a spamszűrőktől elvárható viselkedést mutat: a spamet sokkal könnnyebben fogadja el rendes levélnek, mint fordítva.

A tesztelés tapasztalatait összefoglalva megállapíthatjuk, hogy ha nem követeljük meg a metaspamek kezelését, akkor a szűrő közel 100%-os pontossággal dolgozik, szélsőséges spamdefiníció esetén is 80%-os, de általában 95-100%-os lefedettséget biztosít, így valós használatra alkalmasnak bizonyul.

6.4.1 Metaspam

A *metaspam* fogalma alatt azt értjük, mikor valaki továbbküld egy spamet, esetleg egy-két sor megjegyzéssel kiegészítve. Ez természetesen nem spam, hiszen pont azért küldik tovább, hogy megmutassák a spamet a címzettnek. Persze egy átlagos felhasználó nem küld tovább spameket, a metaspam leginkább éppen a spameléssel foglalkozó kutatókat érinti. Ez kemény dió a statisztikai spamszűrőnek, mert az esetleges kiegészítő szövegen kívül csak a fejlécsorokra hagyatkozhat. *Graham* szerint nem is várható el egy szűrőtől,

hogy ezeket a leveleket automatikusan rendesnek tekintse, egy külön jelszavas módszert javasol, ami biztosítja, hogy az ilyen levelek átjussanak a szűrőn [13].

Témavezetőm továbbküldött nekem érdekességképpen egy levelet, ami spamszűrőket reklámozott. Ez a klasszikus UBE (ld. 2.1.1. fejezet) definíció szerint egyértelműen spamnek tekintendő, témájától függetlenül. Megnéztem, hogy a szűrő különféle paraméterbeállítások mellett hogyan ítéli meg az eredeti levelet, illetve a továbbküldött változatot. Egyetlen esetben jött ki a tökéletes eredmény – ti. hogy az eredeti levél spam, a továbbküldött viszont metaspam lévén rendes levél – az általam javított paraméterezésű bayesi szűrővel (ld. 43. o.). Ez persze lehet véletlen, de mindenesetre biztató.

6.5 Fejlesztői dokumentáció

A fejlesztői dokumentáció olvasása közben javasolt a forrásfájlok egyidejű tanulmányozása. A programlistákat ld. az A. függelékben. Az itt közölt leírás nagyban támaszkodik az 5.1. és a 6.3. fejezetekre.

A szűrő négy fájlból áll:
`SHF.pm`, `SHFlearn.pl`, `SHFapply.pl` és `SHFwrapper.pl`.

Először tekintsük át az általánosan használt konstansok és eljárások definícióját tartalmazó `SHF.pm` fájlt.

A „konstansok” és az „algoritmus paraméterei” rész magáért beszél.

A `tokenize` eljárás végzi a tokenizálást. Többféle tokenizálóval próbálkoztam, végül az a változat bizonyult a legjobbnak, mikor a `.,;:?!` karaktereket leválasztottam a szavakról, egyébként pedig a szóközzel elválasztott nemszóköz-sorozatok lettek a tokenek.

A `kgram2index` eljárás a k-gramot adatbázisindexszé alakítja.

A `write_db` és `read_db` eljárások az adatbázis írását/olvasását valósítják meg. Az adatbázis első két sora a feldolgozott rendes és spamlevelek darabszámát tartalmazza, majd soronként egy k-gramot a hozzá tartozó előfordulási számmal a spamek és a rendes levelek körében.

A `getprob` eljárás számítja ki az adatbázis alapján az egyes k-gramokhoz tartozó *Graham*-féle valószínűséget a következő algoritmus alapján:

ha nincs az adatbázisban a k-gram, akkor 0,4
ha $N_R = 0$, akkor 0,99
ha $N_S = 0$, akkor 0,01
különben $\frac{n^S/N_S}{n^R/N_R + n^S/N_S}$

ahol n^R és n^S az adott k-gram előfordulásainak száma a rendes illetve a

spamlevelek között. A hányadosok *Graham*-féle 1-re normálását elhagytam, és nem szorítottam be az értékeket 0,01 és 0,99 közé.

A `read_mail` eljárás egy mailbox formátumú emailfájlt olvas be és egy tömbben visszaadja a leveleket. A fájl feldarabolását a `split_mails` eljárás végzi az alapján, hogy a fájlban az emailek kezdetét '`\nFrom `', azaz újsor karakter, a '`From`' string és egy szóköz karakter jelzi. Ezenkívül ha van, eltávolítja a Pine-specifikus automata kezdőüzenetet a levelek közül.

Az `SHFlearn.pl` scripttel taníthatjuk a szűrőt. Beolvassuk az adatbázis-fájlt, ha nincs, akkor új nyelvi modellt építünk. A `learn` eljárás segítségével végigolvassuk a leveleket, és az adatbázis-hashben tároljuk az egyes k-gramok előfordulási számát, a rendes levelek-beli előfordulásokat a megfelelő szorzóval számolva (vö. 43. o.). Kiírjuk a teljes adatbázist, majd elhagyjuk a túl kevésszer előforduló k-gramokat, és kiírjuk az alkalmazáshoz szükséges gyakoribb k-gramokat tartalmazó adatbázist.

Az `SHFapply.pl` script valósítja meg a szűrést. Egy egy levelet tartalmazó fájlt olvas be, tokenizálja, kiválasztja a legszélsőségesebb valószínűségű k-gramokat, majd kiszámítja a spamvalószínűséget, *Graham* módszere szerint. Ezt összehasonlítja a küszöbbel és az eredménynek megfelelően `OK`-t vagy `SPAM`-et ír ki.

Az `SHFwrapper.pl` feladata mindössze annyi, hogy a Procmail rajta keresztül hívhatja meg az `SHFapply.pl`-t, aminek eredményét a levél tárgyába és fejlécébe elhelyezi.

6.5.1 Hatékonyság: Perl vagy C++

A 6.3-beli kísérletek lefolytatása meglehetősen sok processzoridőt vett igénybe, ezért gyorsabb működés érdekében próbaképpen a tanítóprogramot megírtam C++-ban is.

A C++ program gyorsabb lett ugyan, viszont több memóriát fogyasztott. Többféle adatszerkezetet kipróbáltam a Perl *hash* kiváltására, de memóriaigényben egyik sem tudta felvenni vele a versenyt. A szövegfeldolgozási – például a tokenizálási – feladatokat is sokkal könnyebben tudtam Perlben kezelni, így végül maradtam az eredeti programnál.

Valószínű, hogy az objektumorientáltság miatt keletkező költség okozta a C++ program nagyobb memóriaigényét, ugyanis a C++ *Standard Template Library*ben minden bizonnyal a lehető leghatékonyabb megvalósító adatszerkezeteket választották. Több tároló, így például a `map` is piros-fekete fákkal van megvalósítva, ami [3] szerint nagyon hatékony.

6.6 Továbbfejlesztési lehetőségek

Fontos lenne a program memóriaigényének csökkentése. Ezt esetleg temporális fájlok segítségével lehet megoldani.

A gyakoribb szavak esetén lehetne lokálisan nagyobb k-val dolgozni. Ehhez viszonylag bonyolult adatszerkezetre lehet szükség: valamilyen faszerkezetre, ami a megfelelő helyeken mélyebb.

Célszerű minél kisebb méretű nyelvi modellt tárolni, legjobb épp akkorát, amekkora a használt k mellett optimális.

Ahogy növekszik az adatbázis, lehet növelni a k értékét is, de jobbnak látszik inkább az adatbázis méretét korlátozni. Ezt úgy tehetjük meg, hogy a régi, szükségtelen bejegyzéseket „elfelejtjük" azaz töröljük a modellből. Ha a szűrő adaptivitását továbbra sem akarjuk elveszíteni, jó megoldásnak tűnik, hogy a nyelvi modell minden bejegyzése mellett nyilvántartjuk, hogy az a k-gram mikor fordult elő utoljára, és a túl réginek ítélt adatokat töröljük. Másik lehetőség, hogy mindig az előző n tanulás eredménye képezze az adatbázist. Ekkor az összes $n-1$ darab régi tanuláshoz tartozó leveleket is tárolni kell, az n-nel ezelőtti adagot pedig folyamatosan törölhetjük. Ezekkel a módszerekkel „kétirányú" adaptivitást érhetünk el: alkalmazkodunk az újhoz és elfelejtjük a régit.

Nagyobb időtávlatban érdemes elgondolkodni a *metatanulás* lehetőségén, azaz hogy a program a tanulási módszerét – például a 6.3. fejezetben szereplő paramétereket – is időről időre fejlessze, változtassa.

7 Összegzés

Dolgozatomban áttekintettem a spamelés problémáját, a spam elleni harc eszközeit, felhívtam a figyelmet néhány fontos aspektusra.

Megvizsgáltam a NBC-re épülő szövegosztályozási módszereket, elemeztem a *Graham*-féle másodikgenerációs statisztikai spamszűrőt, és összehasonlítottam az NBC-s módszerekkel.

Implementáltam és továbbfejlesztettem ezt a szűrőt, és mértem a teljesítményét.

A szógyakoriságokon alapuló szűrő kiterjesztése szó-n-esekre nem hozott sikert, viszont az eredeti szűrőt hatékonyabbá tudtam tenni bizonyos belső paraméterek javításával.

A kifejlesztett spamszűrőt teljesítménye alkalmassá teszi a valós felhasználásra. A programot le lehet tölteni a `http://shf.azbest.hu` címről.

A függelék – programlisták

A.1 Az SHF.pm fájl

```
package SHF;
require Exporter;
@ISA = qw(Exporter);

# --- konstansok ---

# fájlnevek
$BASE_DIR = "$ENV{'HOME'}/.SHF"; # a program könyvtára
$DB_FN = 'DB';                   # az adatbázis-fájl neve
$DB_FULL_FN = 'DB_full';         # a teljes adatbázis-fájl neve

$NODB = 'NODB'; # read_db visszatérési értéke, ha nem találja az adatbázist

$KGRAMSEP = "\002"; # az adatbázisban a k-gram tagjait elválasztó karakter
$FIELDSEP = "\003"; # az adatbázisban a mezőket elválasztó karakter

$OK = 'ok';
$SPAM = 'spam';

# --- az algoritmus paraméterei ---

$K = 1; # k-gramok mérete

$fac = 20; # ennyi tényezőt választunk ki

$OK_FACTOR = 1.5; # a rendes levelekben előforduló szavakat
                  # ezzel a szorzóval számítjuk
$SPAM_FACTOR = 1; # ez mindig 1 !

$MIN_APPEAR = 5; # minimális előfordulási szám

# valószínűségek, ha csak az egyik kategóriában fordult elő az adott k-gram
$MAX_PROB = 0.99;
$MIN_PROB = 0.01;
# valószínűség, ha egyik kategóriában sem fordult elő az adott k-gram
$NOT_SEEN_PROB = 0.4;

$SPAM_THRESHOLD = 0.9; # spamküszöb

# -----

sub tokenize {
  $_[0] =~ s/([.,;:?!])/ $1 /gs;
  return split /\s+/, $_[0];
}
```

```
# -----

sub kgram2index {
  return join "$KGRAMSEP", @_;
}

# -----

sub write_db {
  my %db = %{$_[0]};
  my $fn = defined $_[1] ? $_[1] : $DB_FN;
  open F, "> $fn" or die "Can not write DB. [$!]";
  print F "$db{'n_ok'}\n";
  print F "$db{'n_spam'}\n";
  while ( my ( $kgram, $v ) = each ( %{$db{'data'}} ) ) {
    print F "$kgram$FIELDSEP$$v{$OK}$FIELDSEP$$v{$SPAM}\n";
  }
  close F;
}

# -----

sub read_db {
  my %db = ( 'n_ok', 0, 'n_spam', 0, 'data', {} );
  my $fn = defined $_[0] ? $_[0] : $DB_FN;
  open F, "$fn" or return ( $NODB, $NODB );
  my $first = 1;
  my $second = 0;
  while (<F>) {
    my $s = $_;
    chomp $s;
    if ( $first ) {
      $db{'n_ok'} = $s;
      $first = 0;
      $second = 1;
    } elsif ( $second ) {
      $db{'n_spam'} = $s;
      $second = 0;
    } else {
      my ( $kgram, $ok, $spam ) = split /$FIELDSEP/, $s;
      ${${$db{'data'}}{$kgram}}{$OK} = $ok;
      ${${$db{'data'}}{$kgram}}{$SPAM} = $spam;
    }
  }
  close F;
  return %db;
}
```

```
# -----

sub get_prob {
  my %db = %{$_[0]};
  my $index = $_[1];
  my $v;
  if ( exists ${$db{'data'}}{$index} ) {
    $v = ${$db{'data'}}{$index};
    if ( $$v{$OK} == 0 ) {
      return $MAX_PROB;
    } elsif ( $$v{$SPAM} == 0 ) {
      return $MIN_PROB;
    } else {
      return ( $$v{$SPAM} / $db{'n_spam'} ) /
             ( $$v{$SPAM} / $db{'n_spam'} + $$v{$OK} / $db{'n_ok'} );
    }
  } else {
    return $NOT_SEEN_PROB;
  }
}

# -----

sub read_mail {
  my @fn = @_;
  my @mails = ();

  foreach my $fn ( @fn ) {
    open F, $fn or die "No mail file: $fn. [$!]";
    my $tmp = $/;
    undef $/;
    my $mails = <F>;
    $/ = $tmp;
    close F;

    my @ms = split_mails ( $mails );
    push ( @mails, @ms );
  }

  return @mails;
}

# -----

sub split_mails {
  my $mails = $_[0];

  my @ms = split /(\nFrom )/, $mails;
  for ( my $i = 1; $i < ( scalar @ms / 2 ); ++$i ) {
```

```
    $ms[$i] = 'From ' . $ms[2*$i];
  }
  $#ms = ( scalar @ms - 1 ) / 2;

  # eltávolítjuk a pine-specifikus automata üzenetet
  if ( $ms[0] =~ m/DON'T DELETE THIS MESSAGE -- FOLDER INTERNAL DATA/m ) {
    shift @ms;
  }
  return @ms;
}

1;
```

A.2 Az SHFlearn.pl fájl

```
#!/usr/bin/perl -w

use strict;
use lib "$ENV{'HOME'}/.SHF";
use SHF;

$| = 1;

my %db;

usage() if scalar @ARGV != 2;
usage() if $ARGV[1] ne 'OK' and $ARGV[1] ne 'SPAM';

chdir "$SHF::BASE_DIR";

if ( -e $SHF::DB_FULL_FN ) {
  %db = SHF::read_db( $SHF::DB_FULL_FN );
} elsif ( -e $SHF::DB_FN ) {
  print "No '$SHF::DB_FULL_FN' file.\n";
  %db = SHF::read_db( $SHF::DB_FN );
} else {
  print "No DB file. Creating new DB.\n";
  %db = ( 'n_ok', 0, 'n_spam', 0, 'data', {} );
}

my @newmails = SHF::read_mail ( "$ARGV[0]" );

my $type = $ARGV[1] eq 'OK' ? $SHF::OK : $SHF::SPAM;
print "Learning ... ";
learn ( @newmails, $type );

print "done.\nFull hash size = ", scalar ( keys ( %{$db{'data'}} ) ), "\n";
print "Writing '$SHF::DB_FULL_FN' ... ";
```

```
SHF::write_db ( \%db, "$SHF::DB_FULL_FN" );
print "done.\n";
print "Cutting hash ... ";

# kihagyjuk a túl kevésszer előforduló k-gramokat (Graham)
while ( my ( $key, $val ) = each %{$db{'data'}} ) {
  if ( ${$val}{$SHF::OK} +
       ${$val}{$SHF::SPAM} < $SHF::MIN_APPEAR ) {
    delete ${$db{'data'}}{$key};
  }
}

print "done.\nHash size = ", scalar ( keys ( %{$db{'data'}} ) ), "\n";
print "Writing '$SHF::DB_FN' ... ";
SHF::write_db ( \%db, "$SHF::DB_FN" );
print "done.\n";

# -----

sub learn {
  my @arr = @_;
  my $code = pop ( @arr );
  my $cnt = 0;

  print "#mails " , scalar ( @arr ), "\n";

  foreach my $mail ( @arr ) {
    my @tokens = SHF::tokenize ( $mail );
    print( ++$cnt . '/' . scalar @arr . ' ' );

    my @kgram = ();
    foreach my $t ( @tokens ) {
      push ( @kgram, $t );
      if ( scalar @kgram == $SHF::K ) {
        my $ref = \${$db{'data'}}{SHF::kgram2index(@kgram)};
        if ( not defined $$ref ) {
          $$ref = { $SHF::OK, 0, $SHF::SPAM, 0 };
        }
        ${$$ref}{$code} +=
          ( $code eq $SHF::OK )
          ? $SHF::OK_FACTOR
          : $SHF::SPAM_FACTOR;
        shift ( @kgram );
      }
    }

    if ( $code eq $SHF::OK ) {
      ++$db{'n_ok'};
    }  else {
```

```
      ++$db{'n_spam'};
    }
  }
}

# -----

sub usage() {
  print "Usage:\n";
  print "SHFlearn.pl mailbox-file OK|SPAM\n";
  exit 1;
}
```

A.3 Az SHFapply.pl fájl

```
#!/usr/bin/perl -w

use strict;
use lib "$ENV{'HOME'}/.SHF";
use SHF;

chdir "$SHF::BASE_DIR";

my %db = SHF::read_db();
if ( defined $db{$SHF::NODB} and $db{$SHF::NODB} eq $SHF::NODB ) {
  print $SHF::NODB;
  exit 1;
}

my ( $l ) = SHF::read_mail( "$ARGV[0]" ); # a vizsgálandó levél
my @t = SHF::tokenize( $l );

my @i = choose_interesting_kgrams( @t );
my $p = calculate_spam_prob( map { $$_[1] } @i );

print $p > $SHF::SPAM_THRESHOLD ? 'SPAM' : 'OK';

# -----

sub choose_interesting_kgrams {
  my @i = ();

  my @kgram = ();
  foreach my $t ( @_ ) {
    push( @kgram, $t );
    if ( scalar @kgram == $SHF::K ) {
      my $index = SHF::kgram2index( @kgram );
      my $p = SHF::get_prob( \%db, $index );
```

```
      push( @i, [ $index, $p ] );
      shift( @kgram );
    }
  }

  # minden csak egyszer forduljon elő (Graham)
  my %seen = ();
  foreach my $i ( @i ) {
    $seen{$$i[0]} = $$i[1];
  }
  @i = ();
  foreach my $z ( keys %seen ) {
    push( @i, [ $z, $seen{$z} ] );
  }

  my @j = sort { dif( $$a[1] ) <=> dif( $$b[1] ) } @i;
  my $from = $SHF::fac;
  if ( $from > @j ) { $from = @j; }
  return @j[-$from..-1];
}

# -----

sub dif {
  return abs ( $_[0] - 0.5 );
}

# -----

sub calculate_spam_prob {
  my $sp = 1;
  my $semp = 1;
  foreach $a ( @_ ) {
    $sp *= $a;
    $semp *= (1-$a);
  }
  return $sp / ( $sp + $semp );
}
```

A.4 Az SHFwrapper.pl fájl

```
#!/usr/bin/perl -w

use strict;
use lib "$ENV{'HOME'}/.SHF";
use SHF;

chdir "$SHF::BASE_DIR";
```

```
my $FN = "SHFtmpfile";

undef $/;
my $mail = <STDIN>;

open F, "> $FN" or die "Can not write to $FN. [$!]\n";
print F $mail;
close F;

my $res = `./SHFapply.pl $FN`;

unlink $FN;

if ( $res !~ /NODB/ ) { # ha nincs adatbázis nem csinál semmit
  if ( $res =~ /OK/ ) {
    $res = 'OK';
  } else {
    $res = 'SPAM';
  }
  # eredmény a tárgysorba
  $mail =~ s/Subject:(.*)$/Subject: [SHF:$res]$1/m;
  # eredmény a fejlécbe
  $mail =~ s/(X-SpamHammerFiltered:) -/$1 $res/m;
}

print $mail;
```

A.5 A .procmailrc.SHF fájl

```
PATH=/bin:/usr/bin
MAILDIR=$HOME/mail
LOGFILE=$HOME/procmail.log

:0 f
* ^.*
| formail -A "X-SpamHammerFiltered: -" -a "Subject: -" | $HOME/.SHF/SHFwrapper.pl

# az alábbi 3 sort kikommentezve a spamek a 'spam' folderbe kerülnek
#:0 a
#* ^X-SpamHammerFiltered: SPAM
#spam
```

Tárgymutató

Irodalomjegyzék

[1] Androutsopoulos, I., Koutsias, J., Chandrinos, K. V., Paliouras G. és Spyropoulos C. D.: *An Evaluation of Naïve Bayesian Anti-Spam Filtering.* In proceedings of the 11th European Conference on Machine Learning. Workshop on Machine Learning in the New Information Age. pp. 9-17. 2000.
`http://arxiv.org/PS_cache/cs/pdf/0006/0006013.pdf`

[2] Baróti, Gy., Bognár, J., Fejes Tóth, G. és Mogyoródi J.: *Valószínűségszámítás.* Nemzeti Tankönyvkiadó, Budapest, 1998.

[3] Cormen, T. H., Leiserson, C. E. and Rivest, R. T.: *Algoritmusok.* Műszaki Könyvkiadó, 1997.

[4] Cranor, L. F. és LaMacchia, B. A.: *Spam!* Communications of ACM, 41(8):74-83. 1998.

[5] Crocker D.: *RFC 2142 – Mailbox names for common services, roles an functions.* 1997. május.
`http://www.ietf.org/rfc/rfc2142.txt`

[6] Domingos, P. és Pazzani, M.: *Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier.* Machine Learning, 29:103-130. 1997.

[7] Dunning, T.: *Statistical identification of language.* Technical Report MCCS 94-273, New Mexico State University. 1994. március 10.
`http://www.comp.lancs.ac.uk/computing/users/paul/ucrel/papers/lingdet.ps`

[8] *EarthLink wins antispam injunction.* Associated Press. 2003. május 7.
`http://www.msnbc.com/news/910791.asp?0si=-&cp1=1`

[9] Gauthronet, S. és Drouard, E.: *Unsolicited Commercial Communications and Data Protection.* Európai Bizottsági jelentés. 2001. február.
`http://europa.eu.int/comm/internal_market/en/dataprot/studies/spamstudyen.pdf`

[10] Graham, P.: *Better Bayesian Filtering.* 2003. január.
`http://www.paulgraham.com/better.html`

[11] Graham, P.: *Filters vs. Blacklists.* 2002. szeptember.
`http://www.paulgraham.com/falsepositives.html`

[12] Graham, P.: *A Plan for Spam.* 2002. augusztus.
`http://www.paulgraham.com/spam.html`

[13] Graham, P.: *Plan for Spam FAQ.* 2003.
`http://www.paulgraham.com/spamfaq.html`

[14] Graham, P.: *Spam is Different.* 2002. augusztus.
http://www.paulgraham.com/spamdiff.html

[15] Graham, P.: *Will Filters Kill Spam?* 2002. december.
http://www.paulgraham.com/wfks.html

[16] Hindle, R.: *A SpamBayes Projekt bemutatása.* Linuxvilág, 2003. április. pp. 21-23.

[17] Hu, J., Turin, W. és Brown, M. K.: *Language modeling using stochastic automata with variable length contexts.* Computer Speech and Language, 11:1-16. 1997.

[18] *Internetguruk harcolnak a spam ellen.* 2003. március 3.
http://index.hu/tech/net/spam0303

[19] Kneser, R.: *Statistical language modeling using a variable context length.* In proceedings of ICSLP-96. Workshop on Neural Networks and Stochastic Modeling. 1996. október.

[20] Krueger, K. A.: *The Spam Battle 2002: A Tactical Update.* 2002. szeptember.
http://www.sans.org/rr/email/spam_battle.php

[21] Lewis, D. D.: *Naïve (Bayes) at forty: The independence assumption in information retrieval.* In proceedings of the 10th European Conference on Machine Learning. pp. 4-15. Springer Verlag, Heidelberg, 1998.

[22] Leyden, J.: *US.mil launches Operation Desert Spam.* 2003. január 13.
http://www.theregister.co.uk/content/6/28839.html

[23] Maron, M. E.: *Automatic indexing: An experimental inquiry.* Journal of the ACM (JACM), 8(3):404-417. 1961.

[24] McCallum, A. és Nigam, K.: *A comparison of event models for Naïve Bayes text classification.* In proceedings of AAAI/ICML-98. Workshop on Learning for Text Categorization. pp. 41-48. AAAI Press, 1998. január.

[25] *Mikor szabályos egy elektronikus levél, mikor kell azt visszaélésnek minősítenünk?* 2002.
http://www.cert.hu/ismertetok/aol_spam.html

[26] Monty Python's Flying Circus: *The Spam Sketch.* 1970. december.
http://www.stone-dead.asn.au/tv-series/sketches/fc-25/spam-sketch.html

[27] Pantel, P. és Lin, D.: *SpamCop – A Spam Classificaion & Organization Program.* In proceedings of AAAI-98. Workshop on Learning for Text Categorization. AAAI Press, 1998. január.

[28] Pásztor, M.: *Spam – támadás felhasználók és rendszergazdák ellen.* 1998-2001.
http://www.cert.hu/ismertetok/spam_uj.html

[29] Peng, F. és Schuurmans, D.: *Combinig Naïve Bayes and n-Gram Language Models for Text Classification.* 2003.
http://ai2.uwaterloo.ca/~f3peng/publication/ECIR03.pdf

[30] Peters, T. személyes emailje. 2002. augusztus.
http://mail.python.org/pipermail/python-dev/2002-August/028216.html

[31] Peters, T. személyes emailje. 2002. szeptember 19.
http://radio.weblogs.com/0101454/stories/2002/09/19/timOnNbc.html

[32] Peters, T. személyes emailje. 2002. november.
http://mail.python.org/pipermail/spambayes/2002-November/002153.html

[33] *The public record of the T3 Direct versus Joseph McNicol case.* 2002. november.
http://t3-v-mcnicol.ilaw.com.au

[34] Rennie, J. D. M.: *ifile.* 2003.
http://www.ai.mit.edu/~jrennie/ifile/

[35] Robinson, G.: *A Bayes-féle levélszűrés matematikai hátteréről.* Linuxvilág, 2003. április. pp. 24-27.

[36] Robinson, G.: *Gary Robinson's Rants.* 2002-2003.
http://radio.weblogs.com/0101454/stories/2002/09/16/spamDetection.html

[37] Roumazeilles, Y.: *What is SPAM?* 2002.
http://www.spamanti.net/en/whatisit.htm

[38] Sahami, M., Dumais, S., Heckermann, D. és Horvitz, E.: *A Bayesian Approach to Filtering Junk E-Mail.* In proceedings of AAAI-98. Workshop on Learning for Text Categorization. pp. 55-62. AAAI Press, 1998. január.
ftp://ftp.research.microsoft.com/pub/ejh/junkfilter.pdf

[39] *Spam Defined.*
http://www.monkeys.com/spam-defined

[40] *Spam sender sues.* Herald Sun.
http://www.heraldsun.news.com.au/common/story_page/0,5478,4439777%255E1702,00.html

[41] *SpamAssassin.* 2003.
http://spamassassin.org

[42] *Spamming.* 2003.
`http://www.wikipedia.org/wiki/Spamming`

[43] Stefka, I.: *Törvényre vár a politikai marketing (interjú Péterfalvi Attila adatvédelmi biztossal)* Magyar Nemzet, 2003. május 10.

[44] Taylor, L.: *Naïve Bayes – Machine Learning.* 2002. szeptember 16.
`http://ling.ucsd.edu/~ltaylor/Naive.Bayes.handout.pdf`

[45] *This Bulk Email Software Made National News!* 2003.
`http://www.desktopserver.com`

[46] Watson, M. `1524138` számú cikke a `news.admin.net-abuse.email` hírcsoportból. 2003. május 9.

[47] Yerazunis, W. S.: *Sparse Binary Polinomial Hashing and the CRM114 Discriminator.* 2003. január 20.
`http://crm114.sourceforge.net/CRM114_paper.html`