統計的手法に基づくスパムフィルタの設計と実装

井上 博之

摘要

インターネットが爆発的に成長するにしたがって，スパムメールが急増しつつある．このため，スパムフィルタ，すなわち「スパムメールとそうでないものとを自動的に識別するツール」に対する需要は高い．本論文では，まずこのようなツールが解決しなければならないテキスト分類問題を定式化したあと，スパムフィルタにおいて特徴抽出に用いる代表的な手法のうち a) スパムの持つ特徴を予め定義しておいて推定する手法 および b) そのような特徴を統計的に学習して推定する手法について概略を述べる．次に，統計的手法を用いた単純ベイズ分類器 (naïve Baysian classifier) について，統計的な学習の構成方法および識別問題の解法を述べ，実際の設計について考察する。最後に，単純ベイズ分類器を用いたスパムフィルタを実装し，誤検知率を最適化するための分類器の調整について考察する。

目次

1　スパムフィルタ

1.1　スパムとスパムフィルタ

はじめに，本論文における「スパム」という用語の定義を述べる．なお，合意に達したスパムの定義というものは未だ存在していない [1] ことを注記しておく．

電子メールは一通あたりの費用が非常に小さい情報伝達手段であり，また限界費用がほとんど逓増しないことから，広告業者などにとっては魅力的な媒体となりうる．とりわけ，インターネットの発達した今日では，費用をほとんど掛けずに，大量の送信対象の電子メールアドレスを本人の承諾なく入手することが可能である．

そのようにして，受信者が承諾していないにも関わらず送られてくる不要な電子メールは，一般的にジャンクメール (junk mail)，迷惑メール (unsolicited mail) あるいは **スパムメール (spam mail)** と呼ばれている (他方，スパムでないものは，Spam (豚肉の缶詰) との対比で「ハム (ham)」と呼ばれる).

なお，特定電子メール送信適正化法 [16] では次のように定義されている:

> **第二条 二**　特定電子メール　次に掲げる者以外の個人（事業のために電子メールの受信をする場合における個人を除く。）に対し、電子メールの送信をする者（営利を目的とする団体及び営業を営む場合における個人に限る。以下「送信者」という。）が自己又は他人の営業につき広告又は宣伝を行うための手段として送信をする電子メールをいう。
>
> イ　あらかじめ、その送信をするように求める旨又は送信をすることに同意する旨をその送信者に対し通知した者（当該通知の後、その送信をしないように求める旨を当該送信者に対し通知した者を除く。）
>
> (以下略)

メールを受信した人から見て，あるメールがスパムメールであるかどうかということはその人にとっては顕らかである．人間が行う場合には「このメールは必要」「このメールはスパム」と分類するのは単純作業でしかない．

任意の人にとってスパムと判定できるメールもあり得る:

> そのような例としては，コンピュータウィルスによるメールが挙げられる．通常，ウィルスによるメールはスパムとは分類しないが，spam/ham の二分法の上では spam といえる．

しかしながら一般には，ある人による判別が全ての人に通用するとは限らない — 例え

ば，見たこともない店舗からの広告メールはスパムであるが，よく使う店舗からのセールの案内は必ずしもスパムとはいえない．また，よくあるスパムに投資を求める内容のものがあるが，ベンチャーキャピタリストにとっては必要な情報であることもありうる．

このことから，スパムメールを定義するためには**恣意的な基準を持った受信者**を想定することが必要となる．

以上を踏まえて本論文では，スパムを**「基準となる受信者にとって，恣意的な基準に基いて不要と判断できる電子メール」**と定義付ける．

また，「スパムとそうでないものを自動的に識別するツール」を**スパムフィルタ**と呼ぶ．

1.2 スパムフィルタの現状

1998 年に Pantel ら [6] および Sahami ら [7] がそれぞれ発表した統計的手法に基づくスパムフィルタは，2002 年になって Graham[2] によって再発見され，有効性が確認された．

従来のスパムフィルタが複雑な設定を要求しつつも数 % の検出漏れ (false negative) ないしは 1 % 程度の誤検出 (false positive) を覚悟しなければならないというレベルだったのに対して，Graham の発表した Bayesian Filter はほとんど調整を必要としないにもかかわらず検出漏れ 0.5 %・誤検出 0.03% を謳っており，スパムフィルタの開発者達に (そしてスパム送信者にも) 大きなインパクトを与えた．本格的な実用化が進んだのは昨年に入ってからのことだが，現在では，従来型スパムフィルタの代表格であった SpamAssassin[*1] にも統計的手法が取り入れられるなど，急速に一般的な手法になりつつある．

しかしながら，特に日本語を含めた多言語の環境下で用いる場合には英語単独の環境での性能には程遠い性能しか期待できないなど，幾つかの課題が残されている．

1.3 スパムフィルタの必要性

スパムは，定義上からも受信者にとって迷惑なものにほかならないから，スパムフィルタが電子メールのユーザにとって「あると便利なツール」なのはもちろんのことである．

一方，スパムの問題点は，単に受信者にとって迷惑であることだけでなく，「大量のメールが送信されることでメール送信システムに多大な負担が掛かる」という点にある [4].

[*1] http://www.spamassassin.org/

抜本的対策は，スパムの送信自体を止めさせることである．この目的において，わが国をはじめ各国は法整備に取り組んでおり [9, 16]，送信者を適正に取り締まることで解決を計っているところである．しかしながら，現時点では法による規制は実効性が低いばかりか，新たな問題[*2]を引き起こす可能性が指摘されている [10].

それに対し，受信者側での対策が普及することにより，スパムを送信する行為をコストモデル的に成り立たなくさせ，結果として送信者を根絶することができる [2] という見方が存在する．この見方に従うならば，スパムフィルタは単なる受動的な対策ではなく，能動的な問題解決手段となるはずである．

2　スパムフィルタが解くべき問題

スパムフィルタは，テキストストリームである電子メールから，スパムとそうでないものを識別しなければならない．これはテキスト分類問題として記述することができる．

2.1　分類問題

分類問題とは，多数の属性を持つインスタンスの集合を，有限個のクラスに分類する問題である．

いま，インスタンスが m 個 $(m < \infty)$ の属性を持つとし，任意のインスタンス $x = \left(x^i; i = 1, \ldots, m\right)$ は, 先験的に n 個 $(n < \infty)$ のクラス $y_1, \ldots, y_n$ のいずれかに属しているとする[*3]．ここで, 全てのインスタンスからなる集合 X からクラスの集合 $Y = \{y_j | j = 1, \ldots, n\}$ への写像

$$y = f_{\mathrm{C}}(x), \quad x \in X, y \in Y$$

を**基準関数**と呼ぶことにする．

このとき，分類問題は，訓練例

$$\left\{(x_k, y_k);\, y_k = f_{\mathrm{C}}(x_k), k = 1, 2, \ldots, N_T\right\}$$

[*2] スパム対策の有効な手立ての一つとして，米 VeriSign 社などは「電子メールの暗号認証技術」を提唱していた．しかし法規制が甘いと，スパム送信者が法を遵守した上で認証を申請した場合，認証機関はこれを拒む法的根拠を持たないために認証せざるを得ない．結果として，認証技術そのものが無意味となってしまう．

[*3] クラスが所与でないような分類問題を考えることもできるが，ここではクラスの集合および各インスタンスがどのクラスに属するかは所与としている．

を用いて損失関数

$$\hat{L}(f) = \sum_{k=1}^{N_T} L\big(y_k, f(x_k)\big)$$

が最小となるような**分類器**

$$y = f(x), \quad x \in X, y \in Y$$

を求める最小化問題と定義できる．ただし，$L\big(y, f(x)\big)$ は適当な損失関数である．

スパム識別問題にあたっては，各メールをインスタンスとし，これを 2 つのクラス y_{S} (スパム) または y_{H} (ハム) に分類するような分類問題を考える．このとき，基準関数とは節 1.1 で述べたような各受信者の判断そのものである．

誤差の評価は 2 種類の誤り率

- 検出漏れ (false negative = **FN**): スパムをスパムでないと判定してしまう．
- 誤検出 (false positive = **FP**): スパムでないものをスパムと判定してしまう．

で行なわれている．定性的には後者の方がペナルティを大きくとるべきことは明白だが，これを損失関数の形にして統一的に評価することは一般的に容易ではない．

なお，“インスタンスは有限個の属性を持つ” とだけ述べたが，単なるテキストファイルでしかないメールから属性をどのように取り出すかは，用いる手法によって異なる．これについては次に述べる．

2.2 インスタンス属性の抽出

■特徴を予め定義する手法 スパムは一般的にはきわめて特徴的である．それらの特徴を予め数え上げておくことも，さほど難しくはない:

- 英語や母国語以外の文字コード
- 無闇に HTML が使われている
- 特定のキーワードの出現率が高い
  - 「$」「円」「¥」
  - 商品名
  - Nigeria などのアフリカの地名
- 無意味なランダムな文字列

こういった特徴を数値化して属性とし，学習アルゴリズムを用いて分類器を最適化すればよい．

■統計的に学習する手法　特徴を定義する手法は，スパムが不誠実な人間によって生み出されていることにより，致命的な欠点を抱えている．スパム業者も当然スパムフィルタの存在を知っているので，さまざまな手段でフィルタの裏をかこうとしてくる．すなわち，フィルタがどのような特徴を検出する傾向にあるのかを知った上で，そのような特徴が簡単には検出できないようにすればよい (中には，スクリプト言語を使って暗号化されたコンテンツを動的に復号化して表示するものもある).

そのため，ある程度まで誤り率が低くなってくると，検査すべき特徴をやみくもに増やしても誤り率は必ずしも減少しなくなるばかりか，スパムが変化しようものなら却って誤検知が増えてしまうことすらある．

これに対し Graham[2, 3] は，メールを**トークン** (後述) に分解し，含まれているトークンからメールがスパムである確率を推定するという方法を提案した．分類問題の枠組で言えば，全てのトークンごとにインスタンスの属性が存在し，そのトークンが含まれていれば 1，含まれていなければ 0 を値として持つことになる．

3　単純ベイズ分類器

統計的アプローチを取る分類器の代表例が単純ベイズ分類器 (naïve Bayesian classifier) である．単純ベイズ分類器の学習アルゴリズムのコンセプトは，

> 訓練例から各トークンがスパムに含まれている確率 (トークンスパム確率) を推定し，これに基いて適用例のインスタンスのスパム確率 (結合スパム確率) を推定する．

というものである．トークンスパム確率から結合スパム確率を算出するのにベイズの結合確率を使うため，この名がある．

3.1　ベイズの結合確率

以下では，分類を確率事象のように捉える．n 個の事象 $y_1,\ldots,y_n$ が相互排反事象で，$y_j;\ j=1,\ldots,n$ のうちからただ一つの事象が必ず起こるとすると，x と y_j の結合確率は，ベイズの法則より

$$p\left(x,y_j\right)=p\left(x|y_j\right)p\left(y_j\right)=p\left(y_j|x\right)p(x),$$

$$\therefore p\left(y_j|x\right) = \frac{p\left(x|y_j\right)p\left(y_j\right)}{p(x)}$$

さらに，x が m 個の互いに独立[*4]な事象 x^i の組合わせで表現できるとすると，

$$p\left(x|y_j\right) = \prod_{i=1}^{m} p\left(x^i|y_j\right)$$

であり，一方

$$\begin{aligned} p(x) &= p\left(x, y_j\right) + p\left(x, \overline{y_j}\right) \\ &= p\left(y_j\right)\prod_{i=1}^{m} p\left(x^i|y_j\right) + p\left(\overline{y_j}\right)\prod_{i=1}^{m} p\left(x^i|\overline{y_j}\right) \end{aligned}$$

であるから，結局

$$p\left(y_j|x\right) = \frac{p\left(y_j\right)\prod_{i=1}^{m} p\left(x^i|y_j\right)}{p\left(y_j\right)\prod_{i=1}^{m} p\left(x^i|y_j\right) + p\left(\overline{y_j}\right)\prod_{i=1}^{m} p\left(x^i|\overline{y_j}\right)} \tag{1}$$

となる。

3.2 スパム識別問題への応用

スパム識別問題においては，クラスは 2 つ ($n = 2$)，すなわち $Y = \{y_{\mathrm{S}}, y_{\mathrm{H}}\}$ である．ゆえに，あるメールがスパムである確率 $p\left(y_{\mathrm{S}}\right)$，ハムである確率 $p\left(y_{\mathrm{H}}\right)$ について以下の関係が成り立つ:

$$p\left(y_{\mathrm{S}}\right) + p\left(y_{\mathrm{H}}\right) = 1. \tag{2}$$

いま，あるトークン x^i が (スパムもしくはハムいずれかの) メール中に出現する確率を $p\left(x^i\right)$ とすると

$$p\left(x^i\right) = p\left(x^i|y_{\mathrm{S}}\right)p\left(y_{\mathrm{S}}\right) + p\left(x^i|y_{\mathrm{H}}\right)p\left(y_{\mathrm{H}}\right) \tag{3}$$

である．ここでトークン x^i のスパム確率 (x^i が出現したとき x がスパムである条件付き確率) を

$$\pi(x^i) := \frac{p\left(x^i|y_{\mathrm{S}}\right)p\left(y_{\mathrm{S}}\right)}{p\left(x^i\right)} = \frac{p\left(x^i|y_{\mathrm{S}}\right)p\left(y_{\mathrm{S}}\right)}{p\left(x^i|y_{\mathrm{S}}\right)p\left(y_{\mathrm{S}}\right) + p\left(x^i|y_{\mathrm{H}}\right)p\left(y_{\mathrm{H}}\right)} \tag{4}$$

[*4] この独立性の仮定から，naïve の名がある。

と書くと，x がスパムである事後確率 $p\left(y_{\mathrm{S}}|x\right)$ は式 (1) – (4) より

$$p\left(y_{\mathrm{S}}|x\right) = \frac{\left(p(y_{\mathrm{S}})\right)^{1-m} \cdot \prod_{i=1}^{m} \pi\left(x^i\right)}{\left(p(y_{\mathrm{S}})\right)^{1-m} \cdot \prod_{i=1}^{m} \pi\left(x^i\right) + \left(1-p(y_{\mathrm{S}})\right)^{1-m} \cdot \prod_{i=1}^{m} \Big(1-\pi\left(x^i\right)\Big)}, \tag{5}$$

となる．したがって，訓練例における偏りから $p\left(y_{\mathrm{S}}\right)$ と $\pi(x^i)$ を推定しておくことで，式 (5) を用いて任意のメールの推定スパム確率を求めることができる．なお，計算途中の桁落ちを防ぐため，実装では乗除算を対数に変換して演算する:

$$\begin{aligned} p\left(y_{\mathrm{S}}|x\right) = \frac{1}{1+\exp\Bigg\{ & (1-m)\Big(\ln\big(1-p(y_{\mathrm{S}})\big) - \ln p(y_{\mathrm{S}})\Big) \\ & + \sum_{i=1}^{m}\Big(\ln\big(1-\pi(x^i)\big) - \ln\pi(x^i)\Big)\Bigg\}} \end{aligned} \tag{6}$$

Graham は，式 (2) に加えてさらに仮定

$$p(y_{\mathrm{S}}) = p(y_{\mathrm{H}}) = 0.5 \tag{7}$$

を導入し，式 (5) を

$$p\left(y_j|x\right) = \frac{\prod_{i=1}^{m} \pi\left(x^i\right)}{\prod_{i=1}^{m} \pi\left(x^i\right) + \prod_{i=1}^{m} \Big(1-\pi\left(x^i\right)\Big)} \tag{8}$$

と簡単化している．

4　トークンとスパムフィルタの設計

ここまで，単純ベイズ識別器に学習させる特徴のことを，定義なしに「トークン」と呼んで取り扱ってきた．本節では

- トークンとは何か，
- トークン化する手法にはどのようなものがあるか，および
- スパムフィルタを設計する上でどのように取扱うべきか

について述べる．

4.1 トークン

■トークンとは何か 本論文で言うトークンは，理想的には「最小の意味単位を構成する単語列または単語の組合わせ」を指している．ただし実装上の便宜から，通常は単語がそのまま用いられる．

■トークン化する手法 トークン化する際の手法は，分かち書きの有る言語の場合と無い言語の場合で大きく異なっている．

英語を始めとする多くの言語では分かち書きが行われている．そのため，単語ごとにトークンとすることで，比較的容易にトークン化できる．しかしながら，複合語・慣用句なども区別せずに単語単位にしていることや，文脈を無視していることから，より“自然”なスパムが出現した場合には用を為さなくなると予測されている．

これに対し Yerazunis[11] は，近接する単語を複数組み合わせたものを特徴値として取り扱う「CRM 114 識別器」を実装し，高い性能を得ている．

日本語など分かち書きを行わない言語の場合，トークン化の難しさがボトルネックとなって検知率または速度が上がりにくい．現在提案されている方式には

- bigram (二文字を単位) に展開して分割する．
  例 入力:「電子メールの利用についての良好な環境の整備」
  出力:「電子」「子メ」「メー」「ール」「ルの」「の利」「利用」「用に」
  「につ」「つい」「いて」「ての」「の良」「良好」「好な」「な環」
  「環境」「境の」「の整」「整備」
- 漢字/かな/カナの境界で区切る．
  例 入力:「電子メールの利用についての良好な環境の整備」
  出力:「電子」「メ」「ー」[*5]「ル」「の」「利用」「についての」「良好」
  「な」「環境」「の」「整備」
- 既存の形態素解析器 (茶筌 [14]，MeCab[13] など) を用いて分かち書きして分割する．
  例 入力:「電子メールの利用についての良好な環境の整備」
  出力:「電子」「メール」「の」「利用」「について」「の」「良好」「な」
  「環境」「の」「整備」

[*5] 長音記号「ー」は記号と判定され，カナとは区切られる．

などがあるが，これらの方法の間での定量的な比較はあまり行なわれていない．節 5 では，これらの比較評価を行って形態素解析器による分類の優位性を確認する．

■スパムフィルタ設計にあたって

内容の代表としてのトークン　メールの特徴値としてトークンを採用する際には，そのトークンが，メールの属しているクラスを代表し，かつそのメールの属さないクラスにはほとんど登場しないようなものであることが望ましい．

したがって，メールのスパム確率を推定する際には，メールを構成する全てのトークンを用いるよりも，より“特徴的”な — 中立の 0.5 から遠い値を持つもの = **代表トークン**だけを選んで用いる方が効率が良いという予測が成り立つ．

代表トークンの選び方としては，

- 0.5 からの距離が大きいものから**一定数**のトークンを選ぶ．
- 0.5 からの距離が**閾値** θ より大きいトークンを選ぶ．

の 2 種類が考えられる．本論文では後者の手法を取り，閾値を変えて性能の変化を調べた．

初めて出現するトークンの取扱い　関連して，始めて出現するトークンに対するスパム確率の初期値 π_0 をどのように設定するかが問題となる．

スパムとハムでは**語彙** (既出トークンの種類) に有意な差が存在し，一般にハムの方が語彙が多い (サンプルデータでは，ハムはスパムの約 3 倍の語彙を持つ)．このことからは，新しいトークンには (語彙の差に応じた) ハム寄りの値を設定するのが論理的であるように思える．

ところが，統計的手法に基づくスパムフィルタが登場して以来，対抗して“スパムらしい”単語を偽装 するスパムが出現している (「Viagra」を「V1ägra」とするなど)．そのような偽装も，追加学習が進めば「スパムらしい」値が付与されて淘汰されるものの，訓練例における語彙の比だけに頼って初期値を低く設定しすぎるのは危険である．

そこで本論文では，初期値 π_0 を代表トークンの閾値 θ と組合わせて設定し，性能の変化を調べた．特に，閾値を設定する場合，初期値を除外するような値に設定すれば統計的推測の裏を欠こうとするスパムの戦略を無効にできる一方，「スパムとハムの語彙の差」という重要な統計的情報を取り入れることができなくなる．実験では，この点についてどちらが優勢となるかに特に注目した．

4.2 設計

前節までの考察を元に，スパムフィルタを設計する．スパムフィルタ本体は 3 つのデータベースといくつかのルーチンから構成される．

■データベース

- **`%spam`** — スパムに登場したトークンを数えるデータベース
- **`%ham`** — ハムに登場したトークンを数えるデータベース
  - 忘却つき追加学習の場合は，これらの計数データベースには登場したメールの時刻が配列の形で保持される．
- **`%score`** — トークンについてのスパム確率データベース
  - 忘却つき追加学習を実行する場合は特に，スパム確率の推定の際いちいち計数データベースを参照しにいくのではデータベース検索のオーバーヘッドが大きいため，学習後にキャッシュしておく．

■ルーチン

- **`train`** — 訓練例学習

  1. メール x と属するクラス y の組を受け取る．
  2. メールをトークン化し $\left(x \mapsto \{x^i | i = 1, \ldots, m\}\right)$, 各トークン x^i について, x に登場した回数をクラス y の計数データベース (`%spam`, `%ham`) 中の x^i のエントリに加える．
  3. 各トークン x^i について，式 (4) を用いて $\pi(x^i)$ を推定し，確率データベース `%score` を更新する．

  ただし，式 (4) に用いる結合確率は計数データのみから厳密に求めることが難しいため，次のような近似を用いる:

$$p(x^i|y_{\mathrm{S}})p(y_{\mathrm{S}}) \approx \alpha \min\left(1, \texttt{\%spam}\left[x^i\right] / |\texttt{\%spam}|\right),$$
$$p(x^i|y_{\mathrm{H}})p(y_{\mathrm{H}}) \approx \alpha \min\left(1, w_{\mathrm{H}} \times \texttt{\%ham}\left[x^i\right] / |\texttt{\%ham}|\right).$$

ここで，`%class`$[x^i]$ は x^i のクラス class での出現数，$|\cdot|$ は各クラスの学習済みメールの数，α は正規化パラメータである[*6]．また，誤検出を避けるためのバイアスとして，ハムでの出現数には重み $w_{\mathrm{H}} > 1$ を乗じている (Cf. Graham[2]; 実験 1).

- **`discriminate`** — 新着メールのクラス判別

  1. メール x をトークン化する $\left(x \mapsto \{x^i\}\right)$.
  2. 各トークンのスパム確率 $\pi(x^i)$ を `%score` から検索する．データベースに含まれていないときは $\pi(x^i) = \pi_0$ とする．
  3. 式 (6) を用いて $p\left(y_{\mathrm{S}}|x\right)$ を推定する．ただし，代表トークンの閾値 θ が設定されている場合は，代表トークンとしてトークンの集合
  $$\tilde{x} = \left\{\xi \,\middle|\, \xi \in \{x^i\},\ |\pi(\xi) - 0.5| > \theta\right\}$$
  を選び，$p\left(y_{\mathrm{S}}|x\right) \approx p\left(y_{\mathrm{S}}|\tilde{x}\right)$ として推定する．
  4. $p\left(y_{\mathrm{S}}|x\right)$ が 0.5 以上のとき，x はスパムと判定される．

  ただし，式 (6) に用いる $p(y_{\mathrm{S}})$ は次式によって推定する:
  $$p(y_{\mathrm{S}}) \approx \frac{|\texttt{\%spam}|}{|\texttt{\%spam}| + |\texttt{\%ham}|}$$

また，トークン化のためのサブルーチンとして以下の 3 つを実装する．

- **`bigram`** — bigram による分割.
- **`block`** — 漢字/かな/カナ境界による分割.
- **`morpheme`** — 日本語形態素解析器 MeCab[13] による分割.

- **`estimate`** 実験用に，与えられた訓練例に 5 分割交叉確認法を適用してスパムフィルタの性能を評価するルーチンを用意する．

[*6] クラス y_j の出現トークンの延べ総数を w_j とすると，w_j 個のトークンから 1 つ抽出したトークンが x_i である確率は $\%j\left[x^i\right]/w_j$; 一方，y_j に属するメールには平均 $w_j/|\%j|$ 個のトークンが含まれるから，$\%j\left[x^i\right]/|\%j|$ は y_j に属するメール中に x^i が出現する個数の期待値を与える．すなわち，ここではこの期待値が結合確率に近似的に比例するとみなしている．

1. 訓練例を 5 等分する．いま，分割した訓練例を $C_1, C_2, \ldots, C_5$ と書く．
2. 各 $C_i, i = 1, \ldots, 5$ について
   （a）残り 4 つのサンプルを使って学習する (`train`).
   （b）学習の結果を用いて，C_i のメールを新着メールとして判別させる (`discriminate`).
3. 5 回を通しての平均で評価を付ける．

5 実装と性能評価

5.1 実装と実験環境の構成

前節の設計を, データベースとして GDBM, 実行オブジェクトとして Perl および Emacs Lisp を用いて実装した．その上で，最適化のための課題となる以下の点について，実際のメールを集めたサンプルを使って実験を行い，性能評価を試みた:

- ハム語彙の重み w_{H}
- 式 (7) を仮定するかどうか
- どのトークン化手法を用いるか
- 代表トークンの閾値 θ と初期のトークン確率 π_0

ここで，実験対象としないものについては表 1 の値を使用した．

表 1: 共通の設定

項目	設定
ハム語彙の重み w_{H}	2.0
式 (7): $p(y_{\mathrm{S}}) = p(y_{\mathrm{H}})$	仮定しない
トークン化	`morpheme`
代表トークンの閾値 θ	0.1
初期のトークン確率 π_0	0.4

実験は, 共通の訓練例に対し, パラメータを変えて `estimate` を実行して結果を比較する形で実施した．訓練例/テストデータには付録 A に記したサンプルから無作為抽出したものを用いた．なお，実験は TurboLinux Workstation 8.0 (CPU: AMD 1.7GHz, Memory: 1.0GB, Perl ver. 5.6.1, XEmacs 21.4) 上で行なった．

5.2 評価基準

スパムフィルタを実装し最適化するにあたって，性能評価の基準として次の 3 点を実験により測定した．

誤り率 節 2.1 で述べたとおり 2 種類の指標があり，本質的には両方を同時に最小化する最適化は不可能である．しかし実際には，後に述べるように十分大きな w_{H} を設定することで，誤検出率は任意に調整できる．したがって最適化の観点からは，w_{H} を固定した状態での検出漏れによって性能評価の基準が得られるものとする．

記憶容量 トークン化手法や代表トークンの取り方の違いによって，データベースに必要となる記憶容量に差が生じることが予測できる．そこで，さまざまな使用環境を想定する上での評価基準の一つとして，3 つのデータベースのファイルサイズの合計を MB 単位で記録した．

実行速度 実行速度もまた受信者環境での性能の面で重要な評価基準である．トークン化処理の速度の指標として，メール 1 点あたりの平均処理速度を msec 単位で求めた．

なお，節 1.1 に述べたようにスパムフィルタは受信者を想定して構成されるべきであり，これらの評価基準の間には必然的な重み付けというものは存在しない．よって，損失関数などの形で統一的に扱うことはしなかった．

5.3 実験

■実験 1: 仮定 (7) および重み w_{H} Graham の導入した仮定 (7) および w_{H} の誤検出率調整に対する有効性を調べるため，式 (7) を用いる場合と用いない場合についてそれぞれ w_{H} を変えて誤り率を調べた (訓練例サイズ $N_T = 1{,}000$)．その結果，図 1 および図 2 を得た． この結果から，w_{H} を大きくしていったとき，誤検出は急速に減少するのに対して検出漏れは線形的にしか増加しないことが分かる．また，式 (7) の仮定は，w_{H} に対する誤検出の減少が若干早まるものの検出漏れが 2 倍近く高まっており，適切ではないことが分かる．

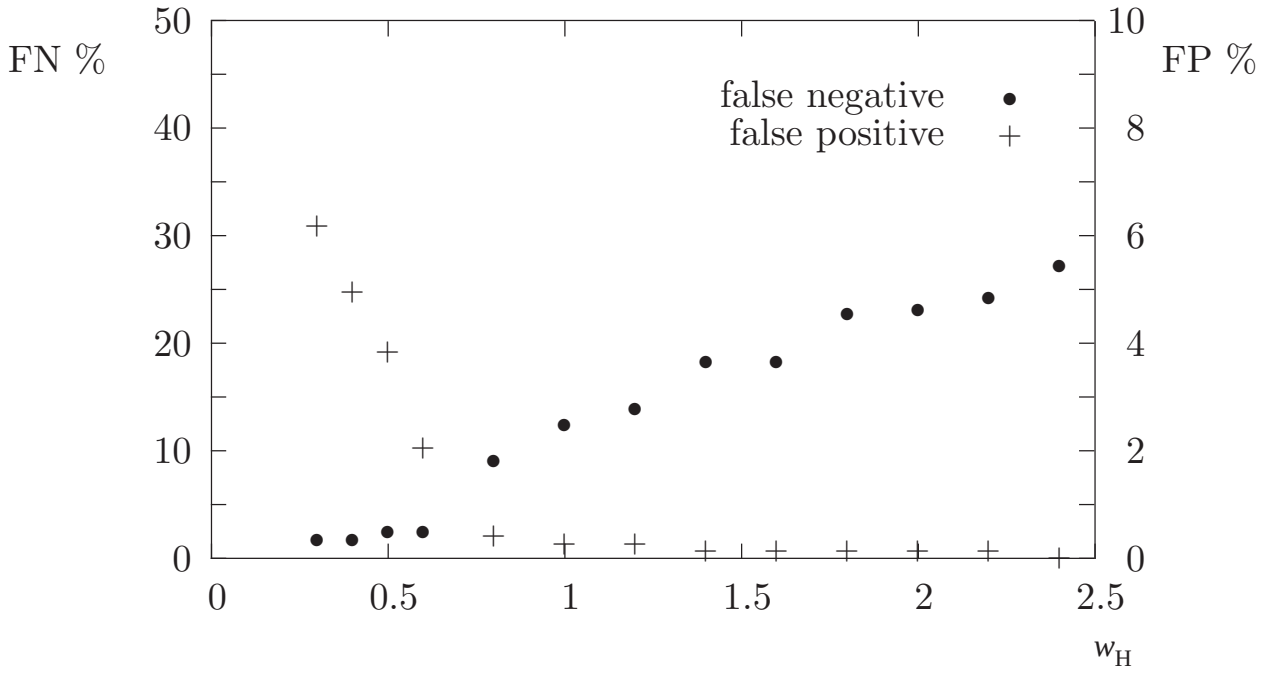

図 1: 重み w_{H} と誤り率 ($N_T = 1{,}000$; 式 (7) を仮定しない)

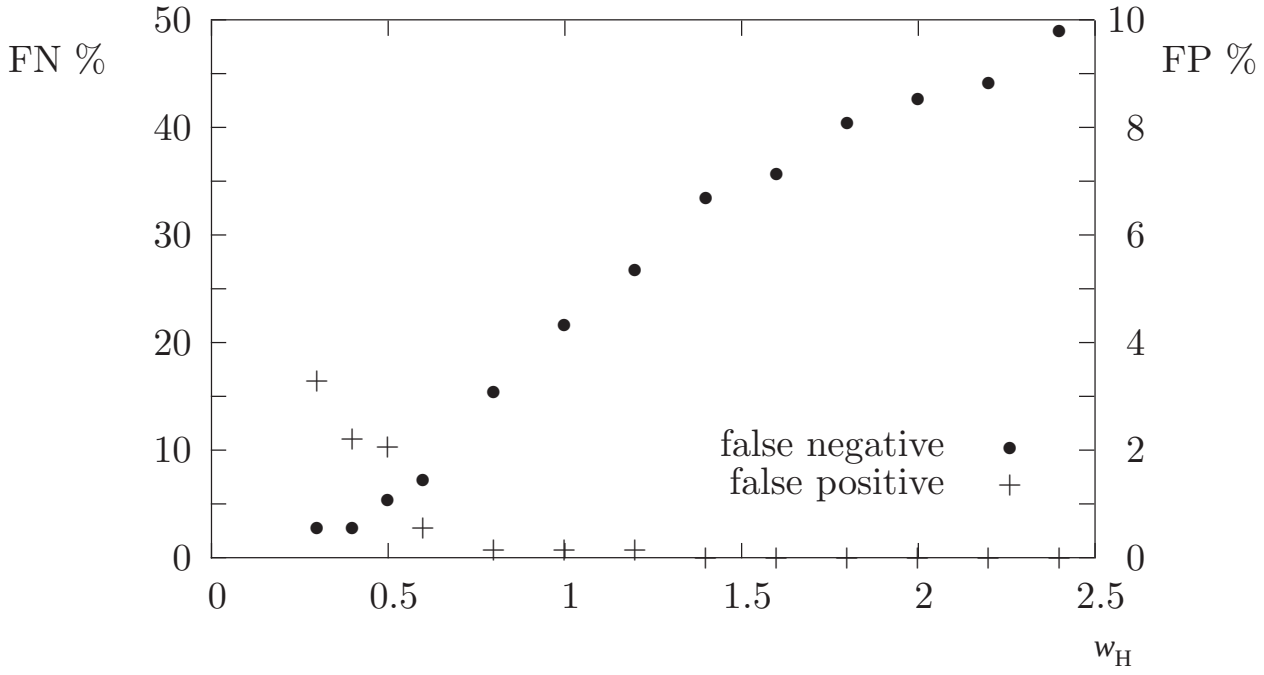

図 2: 重み w_{H} と誤り率 ($N_T = 1{,}000$; 式 (7) を仮定)

■実験 2: 代表トークン閾値 θ および π_0　閾値と初期値を変化させたとき，検出漏れ率がどのように変化するかを調べた．

まず，閾値を $\theta = 0$ に固定して初期値を変えて誤り率を調べたところ，表 2 のような結果を得た ($N_T = 1{,}000$)．この実験結果から，検出漏れ率は π_0 に依存して変化しており，この訓練例については $\pi_0 = 0.4$ 付近で最小となっていることがわかる．

次に，初期値を 0.5 に固定して閾値 θ を変えて誤り率を調べたところ，表 3 のような結果を得た ($N_T = 1{,}000$)．

最後に，初期値を $\pi_0 = 0.4$ に固定して閾値 θ を初期値の付近で変えて誤り率を調べたところ，表 4 のような結果を得た ($N_T = 1{,}000$)．

これらの実験結果から，次のことが言える:

- 検出漏れは θ が小さいほど低く抑えられる．すなわち，たとえスパムが検出を避けるために中立的な語彙を散りばめて偽装していたとしても，多くのトークンを用いた方が正確な判別が可能である．
- 初期値が代表トークンに含まれてしまうほど小さな閾値では，誤検出のリスクが高くなる．

■実験 3: 日本語のトークン化　節 4.1 で検討した 3 つのトークン化手法について比較評価を行った．Ota[5] は，茶筌を用いた実装では

	誤り率	記憶容量	実行速度
bigram	コーパスがある程度大きければ低い	最大	速い
block	やや低い	中程度	速い
morpheme	低い	小さい	遅い

という定性的な結果が得られたとしている．これについて，MeCab を用いたトークン化を実装し，いくつかの訓練例サイズに対して実験を行った (訓練例サイズ $N_T = 1{,}000;\ 5{,}000;\ 20{,}000$)．

その結果，いずれの訓練例サイズでも，表 (5)〜(7) のとおり全ての評価基準について `morphone` が最も良い性能を示した．したがって，形態素解析器を用いる実行速度低下は限定的なものであり，高速な形態素解析器を使うことでより性能の良いスパムフィルタを構成できることがわかった．

表 2: トークン確率の初期値 π_0 と誤り率 ($\theta = 0$, $N_T = 1{,}000$)

π_0	FN %	FP %
0.20	16.6	0.0
0.25	10.6	0.0
0.30	0.755	0.1
0.35	0.604	0.1
0.40	0.049	0.1
0.45	0.0943	0.4
0.50	1.81	0.8
0.55	1.81	0.8

表 3: 代表トークン閾値 θ と誤り率 ($\pi_0 = 0.5$, $N_T = 1{,}000$)

θ	FN %	FP %
0.25	36.5	0.0
0.20	31.1	0.0
0.15	29.9	0.0
0.10	22.1	0.0
0.05	20.9	0.0
0.0	13.1	0.3

表 4: 代表トークン閾値 θ と誤り率 ($\pi_0 = 0.4$, $N_T = 1{,}000$)

θ	FN %	FP %
0.05	8.68	0.3
0.09	9.06	0.3
0.10	23.8	0.0
0.11	24.9	0.0
0.20	31.3	0.0

* $\theta = 0.10$ はちょうど初期値 $\pi_0 = 0.4$ を代表トークンから外す最小の閾値.

表 5: 日本語のトークン化 ($N_T = 1{,}000$)

評価基準		bigram	block	morpheme
誤り率 %	FN	25.2	25.2	24.8
	FP	.000	.000	.000
記憶容量 /MB		9.0	7.0	4.6
実行速度 /ms		71.6	42.4	40.8

表 6: 日本語のトークン化 ($N_T = 5{,}000$)

評価基準		bigram	block	morpheme
誤り率 %	FN	8.06	8.06	7.36
	FP	.000	.000	.000
記憶容量 /MB		26	24	15
実行速度 /ms		81.2	44.3	44.0

表 7: 日本語のトークン化 ($N_T = 20{,}000$)

評価基準		bigram	block	morpheme
誤り率 %	FN	4.82	4.80	4.77
	FP	.047	.047	.047
記憶容量 /MB		59	59	35.8
実行速度 /ms		98.4	49.1	48.7

6 結論

- 統計的手法に基づくスパムフィルタを設計し，実装した．
- Graham の手法のうち，w_{H} を用いて false negative にバイアスを掛ける手法については有効性が確認できたが，スパムとハムが同確率であると仮定することの有効性は確認できなかった．
- 中立なトークンや初出現トークンの扱いについて，誤り率を低くする観点から考察した．
- 日本語のメールのトークン化には，実行速度も含めた全ての面で，形態素解析器を用いた手法が優れていた．

A　実験サンプルについて

ここでは，実験に用いたサンプルについて簡単に述べる．

実験では，サンプルとして，実際に著者が受信したメール (スパム/ハム) から 30,000 点を無作為抽出したものを用意した (サンプル `Original`).

各サンプルに含まれるメールの構成比は表 8 の通り．

表 8: サンプルの構成比

	スパム	ハム
メール数	7,964	22,036
(比率 %)	26.5	73.5
のべトークン数	1,045,912	13,163,110
(内，ユニーク)	186,626	568,937
日本語メール数	106	16,100
〃　　比率 %	1.33	73.1

参考文献

[1] Anti-Spam Research Group: *ASRG FAQ*, http://asrg.sp.am/about/faq.shtml.

[2] Graham, P.: *A Plan for Spam*, http://www.paulgraham.com/spam.html.

[3] Graham, P.: *Better Bayesian Filtering*, http://www.paulgraham.com/better.html.

[4] Hambridge, S. and Lunde, A.: DON'T SPEW — A Set of Guidelines for Mass Unsolicited Mailings and Postings (spam), Technical report, Network Working Group, 1999, RFC 2635, FYI 35.

[5] Ota, S.: *(Scheme) (Lisp)*, http://www.geocities.co.jp/SiliconValley-PaloAlto/7043/.

[6] Pantel, P. and Lin, D.: SpamCop: A Spam Classification & Organization Program, in *Learning for Text Categorization: Papers from the 1998 Workshop*, AAAI Technical Report WS-98-05, 1998, http://www.cs.ualberta.ca/~ppantel/Content/papers.htm.

[7] Sahami, M., Dumais, S., Heckerman, D. and Horvitz, E.: A Bayesian Approach to Filtering Junk E-Mail, in *Learning for Text Categorization: Papers from the 1998 Workshop*, AAAI Technical Report WS-98-05, 1998, http://citeseer.nj.nec.com/sahami98bayesian.html.

[8] SpamAssassin: *SpamAssassin Documentation*, http://www.spamassassin.org/doc.html.

[9] U. S. Federal Law: *Controlling the Assault of Non-Solicited Pornography and Marketing Act of 2003*, 2003.

[10] Wired News: *Tomorrow's Menu: Spam, Spam, Spam*, http://www.wired.com/news/politics/0,1283,61555,00.html.

[11] Yerazunis, W. S.: *Sparse Binary Polynomial Hashing and the CRM114 Discriminator*, http://crm114.sourceforge.net/CRM114_paper.html.

[12] 神崎正英：日本語と文字コード, http://www.kanzaki.com/docs/jcode.html.

[13] 工藤拓：*MeCab: Yet Another Part-of-Speech and Morphological Analyzer*, http://cl.aist-nara.ac.jp/~taku-ku/software/mecab/.

[14] 松本裕治, 北内啓, 山下達雄, 平野善隆, 松田寛, 高岡一馬, 浅原正幸：形態素解析システム『茶筌』version 2.3.0 使用説明書, 2003.

[15] 渡辺澄夫：データ学習アルゴリズム, データサイエンス・シリーズ, 第 6 巻, 共立出版, 2001.

[16] 日本国内法：特定電子メールの送信の適正化等に関する法律, 2002.