知識情報演習Ⅲ（前半第3回）

辻　慶太

http://slis.sakura.ne.jp/cje3

索引付けの手順概要（復習）

(1) 索引語の候補の抽出 extract.pl

文字バイグラム，単語，フレーズなど

(2) 不要語の削除 stopword.pl

(3) 接辞処理 stemming.pl

plays → play
playing → play

(4) 索引語の重み付け tf.pl idf.pl

検索手法（検索モデル）によっては不要

例えば，論理式によるブーリアンモデルでは不要

(5) 索引ファイルの編成

索引語の重み付け
（以下しばらくは，前回のppt と同じ）

- ある文書を特徴付ける索引語には高い重みを与える
- 伝統的な手法に TF.IDF法がある
  - TF: 索引語頻度
  - IDF: 逆文書頻度

- 完全一致（ブーリアンモデル）では不要

TF: 索引語頻度

- Term Frequency（TF） → ここで言うTermとは索引語を表す

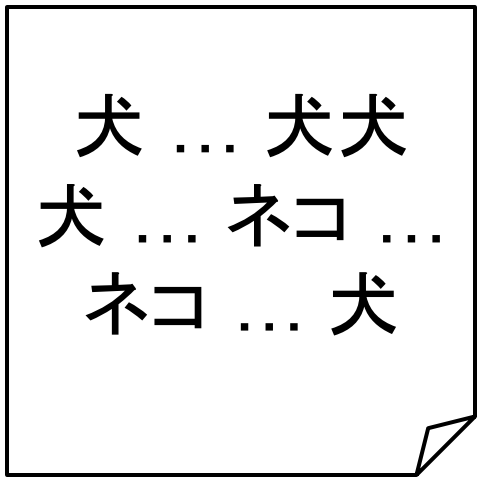
- $tf(t,d)$ と表す。

  文書 d における索引語 t の出現頻度

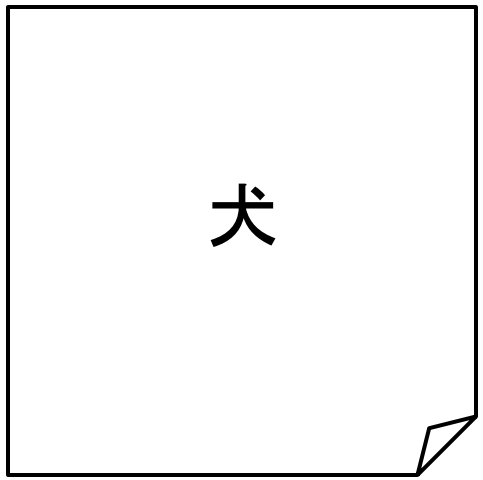
  → なぜ用いるか？

  → ある文書によく出現する索引語は，その文書をよく特徴付けるだろうという仮説に基づく

TFの例



文書A



文書B

$tf(犬, A) = 5$

$tf(ネコ, A) = 2$

$tf(犬, B) = 1$

索引語の重み付け

- ある文書を特徴付ける索引語には高い重みを与える
- 伝統的な手法にTF.IDF法がある
  - TF: 索引語頻度
  - IDF: 逆文書頻度

これから詳細を説明

- 完全一致（ブーリアンモデル）では不要

→ ブーリアンモデルでは索引語に「あるかないか」だけ考える。「どれくらいあるか」は考えない。

IDF: 逆文書頻度

- Inverse Document Frequency（IDF）
- 少数の文書にしか現れない索引語を重視する

$$idf(t) = \log \frac{N}{df(t)} + 1$$

N: コレクション中の文書総数

df(*t*): 索引語 *t* が出現する文書数

→ なぜ用いるか？

→ TFだけでは問題がある。TFが高い語は多くの文書に出現する為，特定の文書を弁別する能力が低い → 例えば「は」「が」などはTFが非常に高いがほとんどどの文書にも現れる為，文書の特徴は表さない（弁別性に欠ける）。

逆文書頻度（つづき）N=100の場合

逆数を取ることで
df(t)が小さいほど
大きな値にする

対数を取ることで変化分
をなだらかにする

1を足して，重みを
正数にする

df(t)	N/df(t)	log(N/df(t))	log(N/df(t))+1
1	100	6.64	7.64
2	50	5.64	6.64
5	20	4.32	5.32
10	10	3.32	4.32
100	1	0	1

IDFの例

動物 ネコ	動物 犬 犬	動物 犬 ネコ	動物 犬 ロボット	動物 動物 犬

$N = 5$

df　動物=5，犬=4，ネコ=2，ロボット=1

~~動物=6，犬=5~~

idf(動物) = 1

idf(犬) = 1.32

idf(ネコ) = 2.32

idf(ロボット) = 3.32

- idfの最小値
- 「動物」では全文書が検索されてしまい，弁別性が低い

Perlにおけるハッシュ

$hash{'dog'}
keys (%hash)

- 配列と違って文字列をキーとして使える
- １つのキーに値を与えたいとき

  例：索引語 dog の IDF が 2.5

  $idf{’dog’} = 2.5;

- 複数のキーの組合せに値を与えたいとき

  例：索引語 dog の文書D001における TF が 10

  $tf{’dog’}{’D001’} = 10;

キーが1つの場合

$idf{key}

%idf

key	value
dog	2.5
cat	1.6
⋮	⋮
year	3.3
⋮	⋮

```
%idf =
('dog' => 2.5,
 'cat' => 1.6,
 'year' => 3.3);
```

```
$idf{'dog'} = 2.5;
$idf{'cat'} = 1.6;
$idf{'year'} = 3.3;
```

Perlにおけるハッシュ

- 配列と違って文字列をキーとして使える
- 1つのキーに値を与えたいとき

  例: 索引語 dog の IDF が 2.5

  $idf{'dog'} = 2.5;

- 複数のキーの組合せに値を与えたいとき

  例: 索引語 dog の文書D001における TF が 10

  $tf{'dog'}{'D001'} = 10;

キーが複数の場合

%tf

key	value
dog	●
cat	●
⋮	⋮
year	●
⋮	⋮

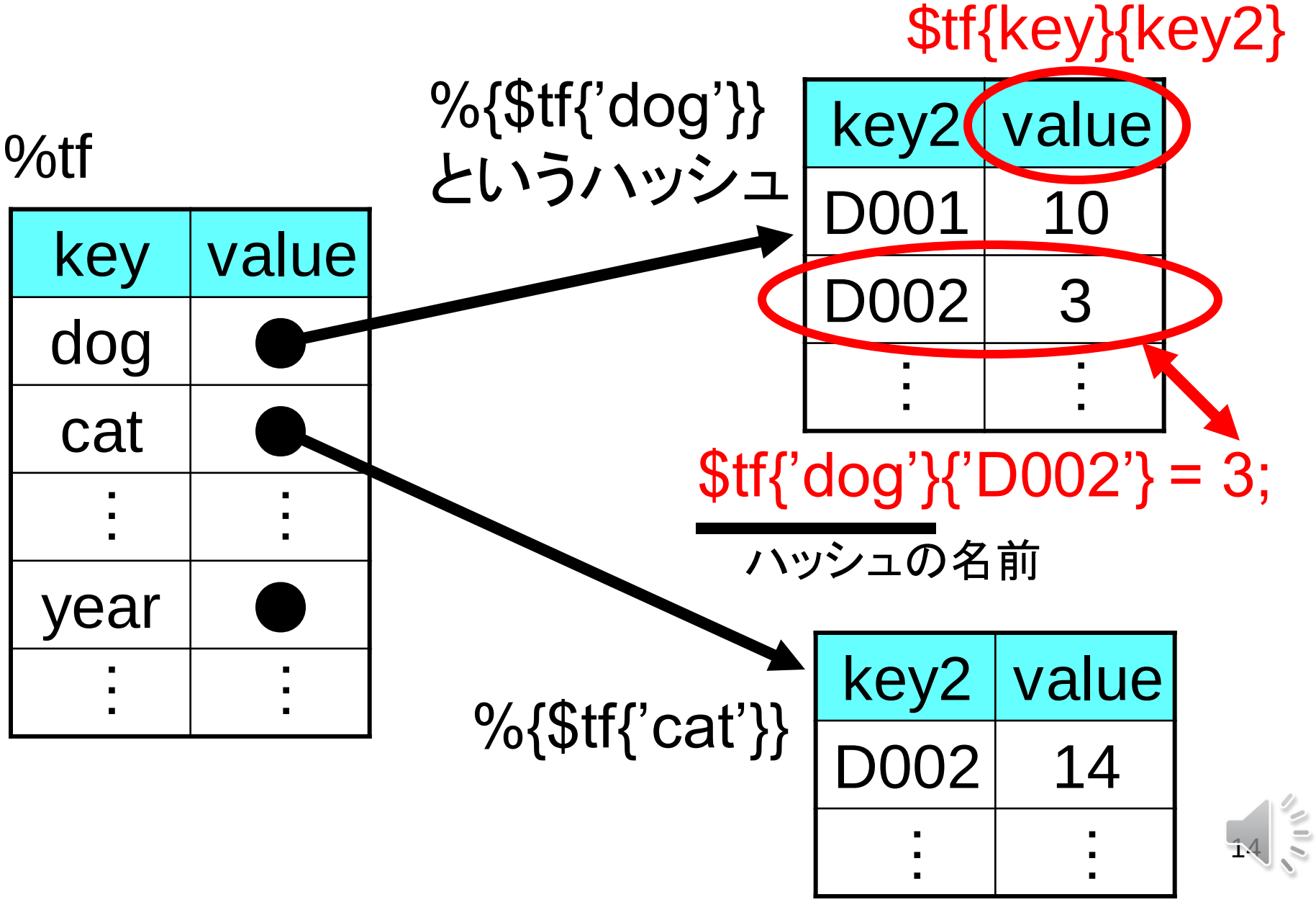
%{$tf{'dog'}}
というハッシュ

key2	value
D001	10
D002	3
⋮	⋮

%{$tf{'cat'}}
というハッシュ

key2	value
D002	14
⋮	⋮

キーが複数の場合


$tf{key}{key2}
%{$tf{'dog'}}
というハッシュ
%tf
key
value
dog
cat
year
key2
value
D001
10
D002
3
$tf{'dog'}{'D002'} = 3;
ハッシュの名前
%{$tf{'cat'}}
key2
value
D002
14

キーが複数の場合

%tf

key	value
dog	●
cat	●
⋮	⋮
year	●
⋮	⋮

%{$tf{'dog'}}
というハッシュ

key2	value
D001	10
D002	3
D005	7

この知識がどう役に立つかというと，先ほど少し述べたように，キーをすべて取り出したい時に役立つ。例えば，dog という語が現れている文書の番号がすべて知りたい時は，

keys %{$tf{'dog'}}　とすれば分かる。

%{$tf{'dog'}} は dog に関するハッシュなので，「そのキーをすべて出せ」と命令すれば，D001，D002，D005... などがすべて出てくる。

ハッシュの内容を出力するプログラムの例

キーが1つ

```
foreach $term (sort keys %idf) {
    print ”$term $idf{$term}¥n”;
}
```

```
dog  2.5
cat  1.6
. . .
year  3.3
. . .
```

キーが2つ

```
foreach $x (sort keys %{$tf{’dog’}}) {
    print ”$x $tf{’dog’}{$x}¥n”;
}
```

```
D001  10
D002  3
. . .
```

演習1

- まず，授業ページにある tf_idf.pl の内容を入力して実行せよ
  - コピーペーストできないPDFファイルなので，全て自分で入力すること
    - その方がプログラムをよく読むでしょう
    - 印刷はできます
- 次に，重み tf(t,d)×idf(t) を計算して出力するように修正せよ
  - 実際には，最後の方に何行か追加すればよい

索引付けプログラムの実装：方針

- 索引付けの段階ごとにプログラムを作る
- 小さめのプログラムを複数作ることで，実装を段階的に行う
  - 大きなプログラムを作ると，中間データの保存が煩雑になる
  - うまく動かない場合に問題の所在が分かりづらい
- 複数のプログラムを連結させる方法
  - 方法１：中間ファイルを作る
  - 方法２：パイプライン処理を行う

索引付けの手順概要（復習）

(1) 索引語の抽出　extract.pl

　文字バイグラム，単語，フレーズなど

(2) 不要語の削除　stopword.pl

(3) 接辞処理　stemming.pl

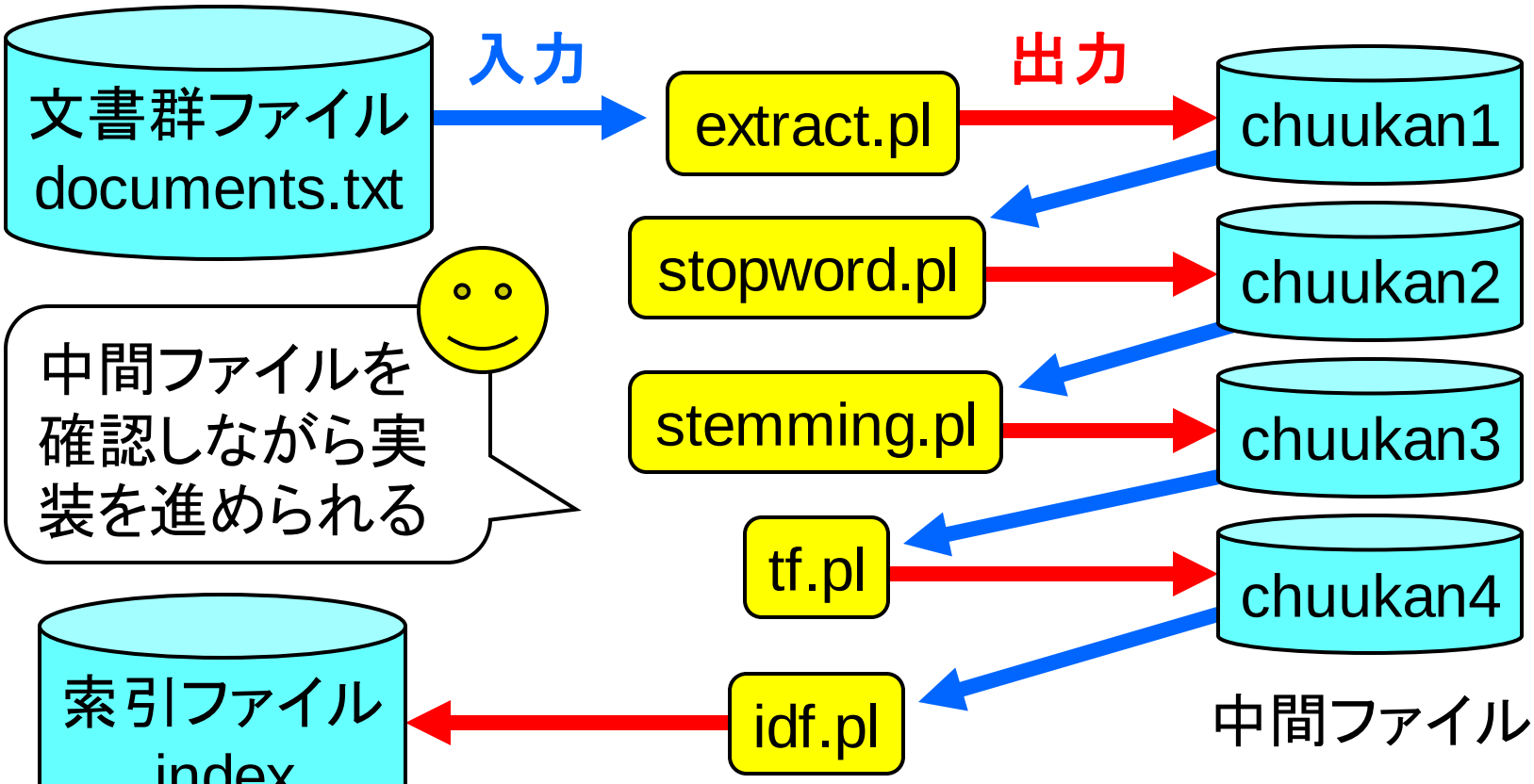
(4) 索引語の重み付け　tf.pl　idf.pl

　検索手法（検索モデル）によっては不要

　例えば，論理式によるブーリアンモデルでは不要

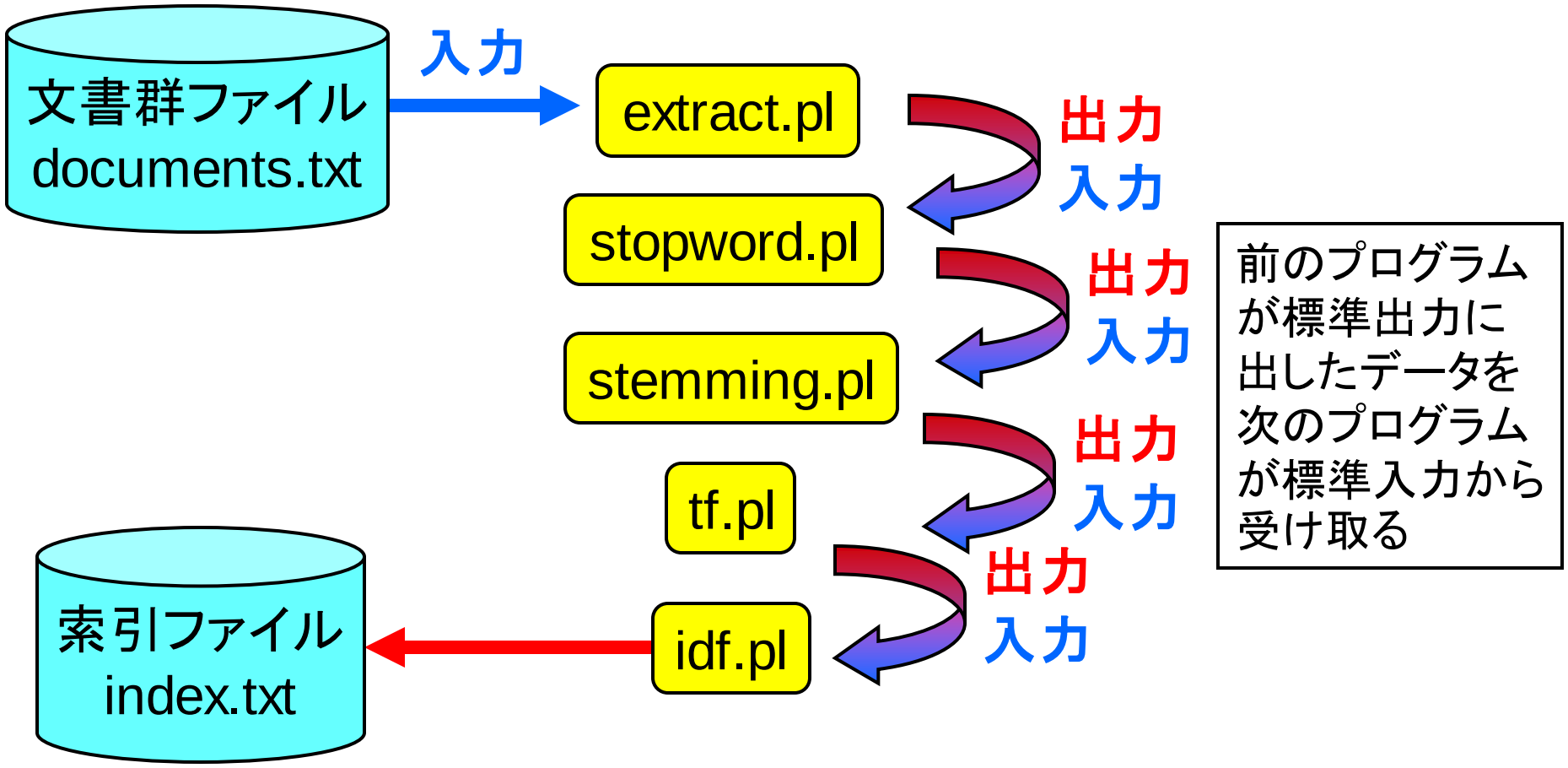
(5) 索引ファイルの編成

連結方法1：中間ファイルを作る




```
% perl extract.pl documents.txt > chuukan1
% perl stopword.pl chuukan1 > chuukan2
% perl stemming.pl chuukan2 > chuukan3
% perl tf.pl chuukan3 > chuukan4
% perl idf.pl chuukan4 > index
```

連結方法2：パイプライン処理を行う



複数のコマンドを縦棒でつなぐ（改行せずに1行で書く）

```
% perl extract.pl documents.txt | perl stopword.pl |
perl stemming.pl | perl tf.pl | perl idf.pl > index.txt
```

参考

- パイプライン処理を行い，かつ中間ファイルも作るには，「tee」というコマンドを間に挟む

% perl extract.prl documents.txt | tee chuukan1 |
perl stopword.prl …（以下，略）

上の例では，chuukan1 というファイルができる

文書群ファイルの形式

```
<DOC>
<NUM>D001</NUM>
<TEXT>
He is a student. …
Students are … student …
She is not a student. …
</TEXT>
</DOC>
<DOC>
<NUM>D002</NUM>
<TEXT>
Two dogs are … The dog is …
</TEXT>
</DOC>
…
```

<DOC> 1つの文書
<NUM> 文書番号
<TEXT> 本文

英文の文書を対象とする

演習のページにある
documents.txt を使うとよい

必要に応じて小さい(または大きい)ファイルを自分で作成してもよい

extract.pl の仕様

- 文書群ファイルを入力し，空白を区切りとして索引語を抽出する
- 索引語を小文字に統一する
- 索引語の末尾に付いたカンマとピリオドを削除する
- 以下の形式で出力する

```
D001  he
D001  is
D001  a
D001  student
…
D002  two
D002  dogs
```

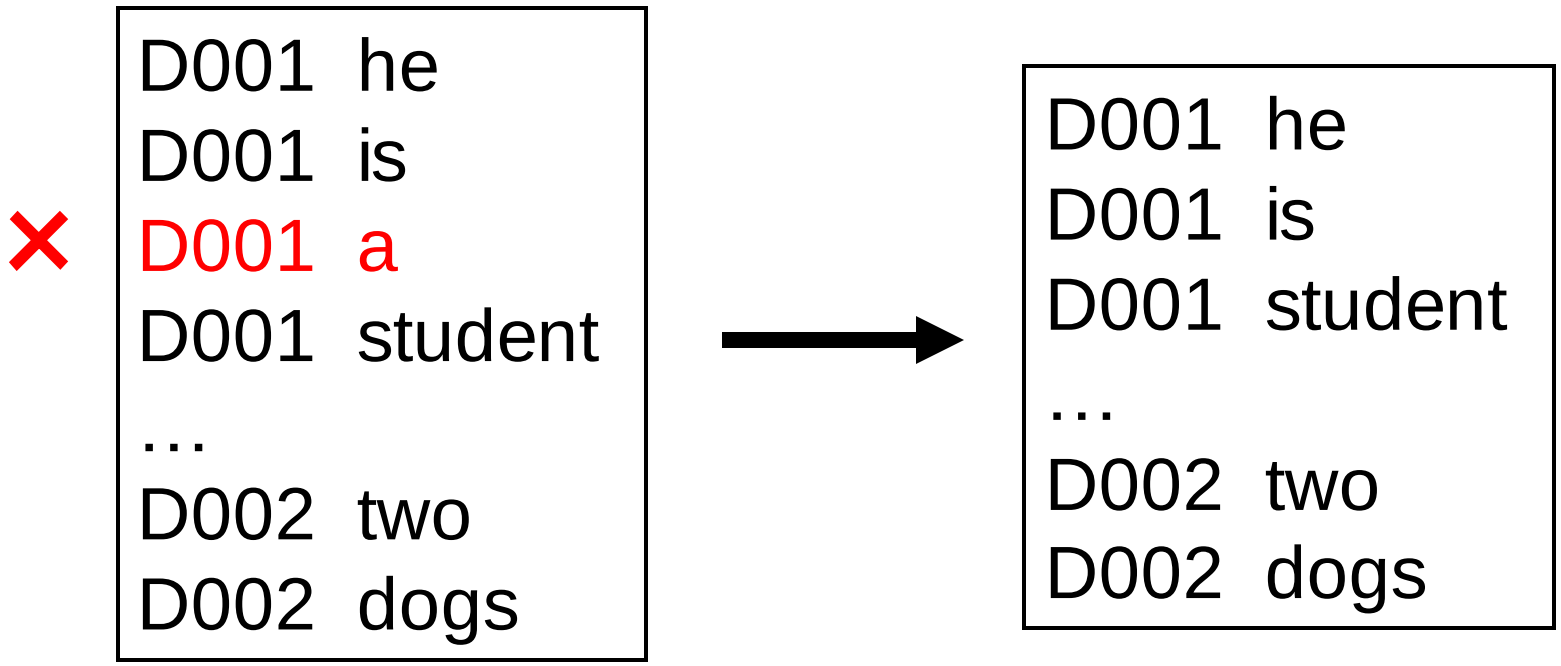
1行に「文書番号　索引語」

文書番号と索引語は半角スペース1つで区切る

stopword.pl の仕様

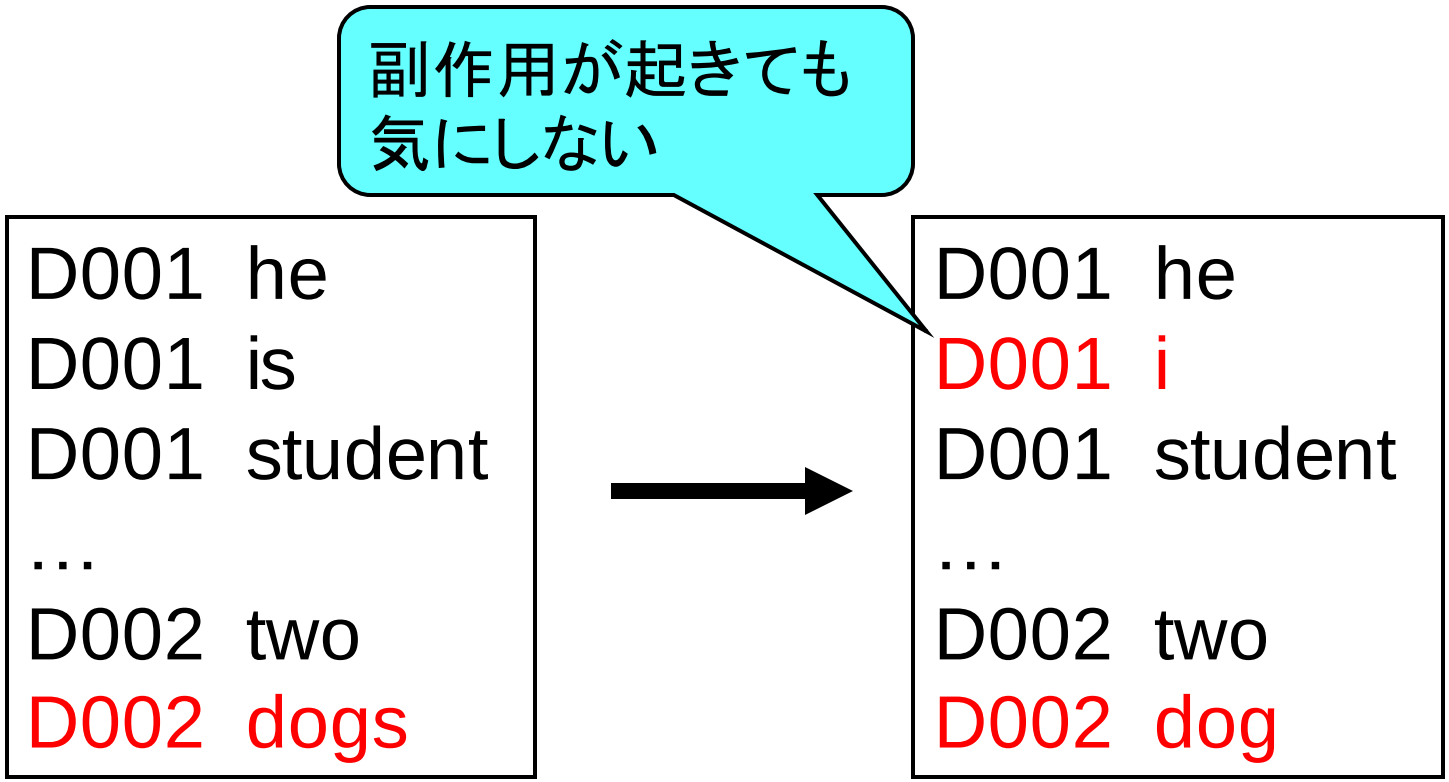
- extract.prl の出力を入力し，不要語を削除する
- 不要語のリスト（自分で適宜追加してよい）

  a, an, and, in, of, the



stemming.pl の仕様

- stopword.pl の出力を入力し，接辞処理を行う
- 接辞処理の規則（自分で適宜追加してよい）
  - 複数形への対応（末尾の s や es を削除）
  - 過去形への対応（末尾の ed を削除）



tf.pl の仕様

- stemming.pl の出力を入力し，文書ごとに索引語の頻度（TF）をかぞえる
- 文書総数をかぞえてファイルの先頭行に出力する

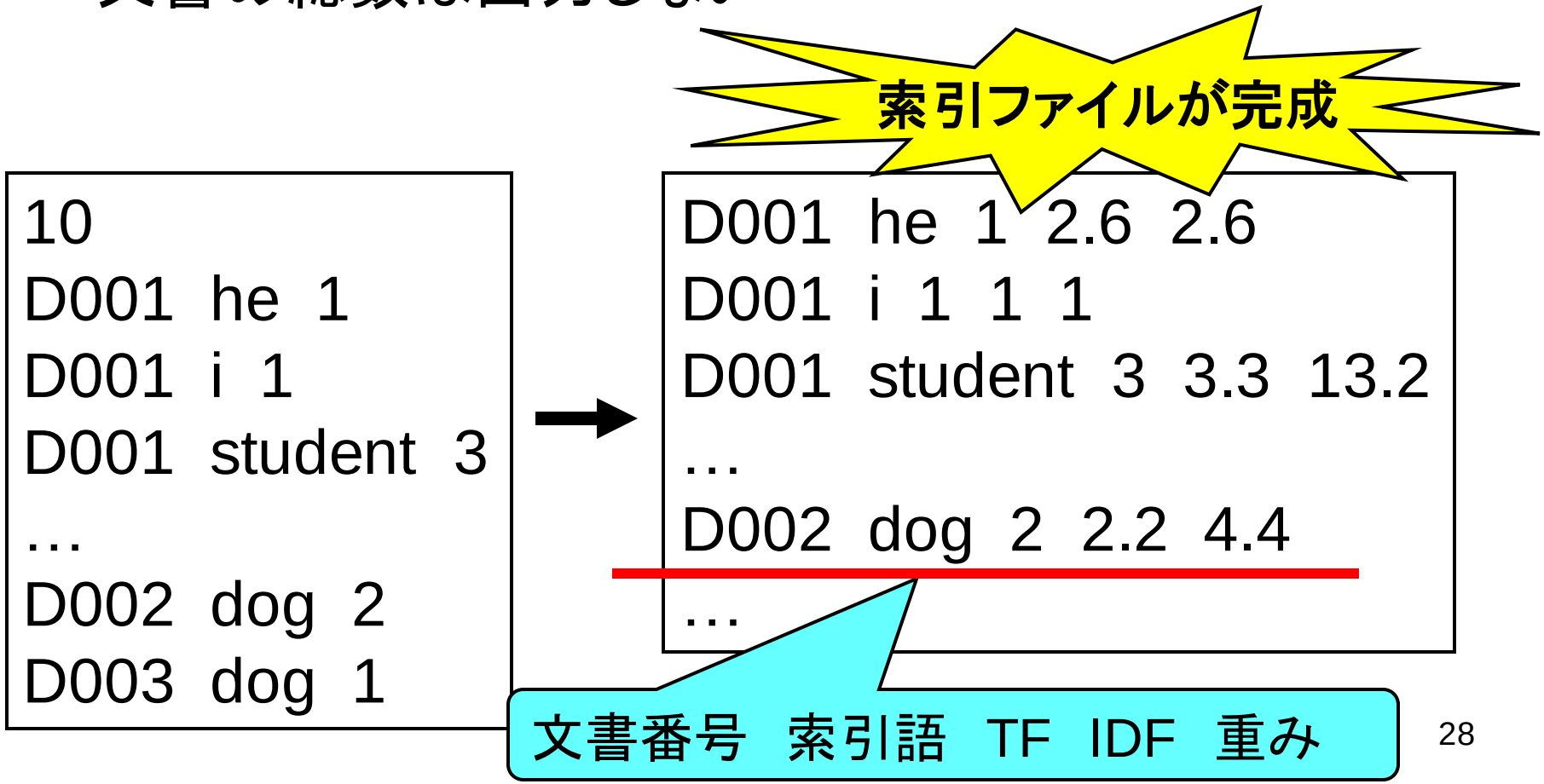
```
D001  he
D001  i
D001  student
D001  student
D001  student
D002  dog
D002  dog
D003  dog
```

文書の総数
（IDFの計算に必要）

```
10
D001  he  1
D001  i  1
D001  student  3
D002  dog  2
D003  dog  1
```

idf.pl の仕様

- tf.pl の出力を入力し，索引語のIDFを計算する
- TF×IDFによって索引語の重みを計算する
- 文書の総数は出力しない



演習2

- extract.pl から idf.pl まで一通り実装せよ
  - 文書数や1文書の長さが異なるいろいろな文書群ファイルを使って動作確認せよ
- 終わった人は:
  - 前半第2回のスライドを参考にして，オンライン処理のプログラムを実装せよ
  - 今回作成した extract.pl，stopword.pl，stemming.pl をそのまま使えるように工夫せよ