言語モデルの作成

目 次

第 1 章　納入モデル、ツール一覧 *1*

第 2 章　日本語基本統計的言語モデル概要 *3*

2.1　構築手順 *3*

2.2　不要部分の除去 *3*

2.2.1　文章でない記事や段落の排除 *3*

2.2.2　括弧 *4*

2.2.3　注視記号 *5*

2.3　文への分割 *5*

2.4　形態素解析 *6*

2.5　形態素の正規化 *7*

2.6　出現頻度の計量 *8*

2.7　高頻度語彙の構築 *9*

2.8　単語 N-gram の計量 *10*

2.9　制限語彙 N-gram の構築 *11*

2.10　バックオフ N-gram モデルの構築 *11*

付 録 A 高頻度語 6 万語言語モデル *13*

A.1　言語モデル及び、音声認識辞書の作成手順 *13*

A.1.1　学習用テキストの作成 *13*

A.1.2　N-gram 言語モデルの構築 *13*

A.1.3　音声認識辞書の作成 *14*

付 録 B CMU-Cambridge 統計的言語モデルツールキット *15*

B.1　はじめに *15*

B.2　ファイル形式 *16*

B.3　言語モデルの作成の作成と評価 *17*

B.3.1 言語モデルの作成 17
B.3.2 言語モデルの評価 22
B.3.3 その他のコマンド 24
B.4 プログラムからの toolkit の利用 26
B.4.1 必要なファイル 26
B.4.2 使用例 26
B.4.3 各関数の解説 27

付 録C N-gram パラメータ削減プログラムのアルゴリズム概要 30
C.1 エントロピーに基づく逐次削減手法 30
C.1.1 back-off 係数の更新法 31
C.1.2 エントロピー変化量 32

付 録D N-gram パラメータ削減プログラム使用説明書 34
D.1 はじめに 34
D.2 インストール方法 34
D.3 オプション説明 36

付 録E 単語正解率判定ツール説明書 37
E.1 データフロー 37
E.1.1 ファイルの準備 38
E.1.2 アラインメント 38
E.1.3 スコアリング 38
E.2 正解・仮説単語列のファイルフォーマット 38
E.3 各プログラムの説明および使用法 40
E.3.1 mkhyp.pl 40
E.3.2 align.pl 40
E.3.3 score.pl 41

付 録F 音声認識辞書作成ツール説明書 42

付 録G 数字表現正規化ツール説明書 45

第1章　納入モデル、ツール一覧†

日本語基本統計的言語モデルは、大語彙連続音声認識のための言語モデルである。語彙のサイズが 5000 語 (5k)、20000 語 (20k)、60000 語 (60k) の 3 種類の言語モデルがある。以下に構成要素を示す。

20k:

20k.htkdic	認識用単語辞書（HTK 形式,21322 エントリ）
20k.vocab	N-gram 語彙辞書　　（45ヶ月分の高頻度順）
45.wit.1.binlm.gz	単語 2-gram　　（45ヶ月,cut-off: 1）
45.wit.4.binlm.gz	単語 2-gram　　（45ヶ月,cut-off: 4）
45.rev.wit.1-1.binlm.gz	逆向き単語 3-gram　（45ヶ月,cut-off: 1,1）
45.rev.wit.4-4.binlm.gz	逆向き単語 3-gram　（45ヶ月,cut-off: 4,4）
45.rev.wit.1-1.10p.arpa.gz	〃　（圧縮版）（45ヶ月,cut-off: 1,1）
75.wit.1.binlm.gz	単語 2-gram　　（75ヶ月,cut-off: 1）
75.wit.4.binlm.gz	単語 2-gram　　（75ヶ月,cut-off: 4）
75.rev.wit.1-1.binlm.gz	逆向き単語 3-gram　（75ヶ月,cut-off: 1,1）
75.rev.wit.4-4.binlm.gz	逆向き単語 3-gram　（75ヶ月,cut-off: 4,4）
75.rev.wit.1-1.10p.arpa.gz	〃　（圧縮版）（75ヶ月,cut-off: 1,1）

5k:

5k.dic	認識用単語辞書 (HTK 形式)
5k.vocab	N-gram 語彙辞書
75.5k.0.wit.binlm.gz	単語 2-gram(cutoff: 0)
75.5k.1.wit.binlm.gz	単語 2-gram(cutoff: 1)
75.rev.5k.0-0.wit.binlm.gz	逆向き単語 3-gram(cutoff: 0,0)
75.rev.5k.1-1.wit.binlm.gz	逆向き単語 3-gram(cutoff: 1,1)

† 伊藤 克亘 (電子技術総合研究所 知能情報部)

75.rev.5k.1-1.10p.wit.arpa.gz	上記の圧縮版 (3-gram 数->10%)

60k:

75.60k.htkdic	認識用単語辞書（HTK 形式，63534 エントリ）
75.60k.vocab	N-gram 語彙辞書　（75ヶ月分の高頻度順）
75.60k.wit.1.binlm.gz	単語 2-gram　（75ヶ月分,cut-off: 1）
75.60k.rev.wit.1-1.binlm.gz	逆向き単語 3-gram（75ヶ月分,cut-off: 1-1）
75.60k.rev.wit.1-1.10p.arpa.gz	〃　10%圧縮版
75.60k.wit.1-1.binlm.gz	前向き単語 3-gram（75ヶ月分,cut-off: 1-1）
75.60k.wit.1-1.10p.arpa.gz	〃　10%圧縮版

言語モデル構築用のツールの構成要素を以下に示す。

chasen-2.02	形態素解析プログラム茶筅
chawan-2.06	形態素読み修正プログラム茶腕
postprocess-1.22	形態素解析後処理プログラム
suuzi_syori-1.0	数字表現正規化プログラム
LMcompress.tar.gz	言語モデル圧縮プログラム

第2章　日本語基本統計的言語モデル概要†

2.1 構築手順

実際に大語彙連続音声認識用の言語モデルを構築するためには、以下のような手順が必要になる。

1. 文中の不要部分の削除
2. 文に分割する
3. 形態素解析
4. 形態素の正規化
5. 高頻度語彙と認識用辞書の構築
6. 単語 N-gram の計量
7. 制限語彙の N-gram の構築
8. バックオフ N-gram モデルの構築

以下、これらの手順について説明する。

2.2 不要部分の除去

統計モデルの学習には、毎日新聞 CD-ROM 1991 年版から 1994 年版までを利用した。新聞記事テキストには、声に出して読むのが難しい様々な表現が含まれることがある。音声認識用の言語モデルを学習するためには、それらの表現を除去した方がよいと考えられる。

2.2.1 文章でない記事や段落の排除

たとえば、新聞データには、記事や段落自体が文章でないものがある。記事や段落自体が文章でないものとしては、人事情報、株式市況などの表、俳句 / 川柳 / 和歌の欄、料理欄の材料一覧、スポーツの結果、アンケート結果などがある。

† 伊藤 克亘 (電子技術総合研究所 知能情報部)

◇ 総合商社大手 6 社の９４年９月中間連結決算

	売上高
三菱商事	８６，６８２（　１．２）
三井物産	８４，２４７（▼ ４．８）

図 2.1. 文章でない段落の例

これらの段落の多くに共通な特徴として、段落全体にひとつも句点「。」が含まれないことがあげられる。そこで、これらの段落を機械的に判別する手法として、句点のない段落を排除する方法が考えられる。

表 2.1. 括弧の用法

	「 」	( )	【 】	“”	『 』	< >	《 》	[ ]
1	○				○			
2	○			○	○	○	○	
3		○						
4		○						
5			○			○	○	○
6		○				○		
7								○

2.2.2 括弧

括弧で囲まれた部分には読まなければ意味が通じないもの (引用句や強調の括弧) から、読むと自然な流れを妨げるものまで様々な用法がある。ある新聞における括弧の用法は以下の通りに分類できる。

1. 引用句 – 読む
   (例)「野球はダサく、サッカーはナウい」と映る。
2. 強調 – 読む
   (例) 今回の調査による「親しみ度」の逆転は、これを明確に示した。
3. リストのラベルの代名詞表現 – 読む
   (例) 同社のカード利用者の大半は契約の際に(１)は了解済み。

4. リストのラベル – 読んでも読まなくてもよい
   (例) 一方、政治家は(１)政治家一九％(２)企業経営者一七％――の順で、自らの役割を高く評価した。
5. 段落などの見出し – (文章と一緒には) 読まない
   (例)　《作り方》　(１) ミズナは長さ 3 センチに切る。
6. 補充要素 – 読まない方が自然
   (例) 一日午前零時、世界最大の砂時計 = 写真 = が始動。

個々の括弧の用法を、表 **2.1** に示す。(○ の箇所は、その用法があることを示す)

表 **2.1** には掲載していないが、新聞では括弧でない記号「 = 」が補充要素を示す括弧的な役割として多用される (上記の **6** の例文参照)。ただし「 = 」は通常の括弧と違って表現の開始と終了が明示的ではないうえ、開始だけ「 = 」で示して、終了は「、」または「。」で示す用法もあるため、自動判別できる範囲が狭い。

削除すべき表現と削除すべきでない表現の両方の用法を持つ括弧が、( ) 〈 〉、< >、《》である。このうち、( ) については、表現の前後の形態素を考慮することで、用法を判別する手法が提案されている [1]。また、< >、《》のうち、見出しの用法とそれ以外は、テキスト中の位置と直後の空白の有無で判別することができる。

これらの自動判別法により、読まない表現である見出しと補充要素を排除する。

2.2.3 注視記号

○ ● ★などの記号は、空白などの字下げ表現と併用して段落の見出しを構成する。図 **2.2** の例文では、「　◇ 高齢化社会をよくする女性の会講演会　」の部分がそれに当る。

空白は句点とともに用いて段落内の文の区切りになるので、空白の前が句点の場合には、文の可能性もある。段落の見出しを構成する場合は見出し全体を削除し、単なる区切りや開始を示す場合には、記号のみを削除する。

2.3 文への分割

日本語では文の区切りは、一般に句点「。」で示す。毎日新聞では、「。」の他に、「？」「！」「。」の後に空白があるものが句点相当の表現として使われている。また記事の種類によって、「▼ 」や「▲ 」が句点および読点相当の表現として使われることがある。

また、句点であっても、たとえば、

> 首相の答えは「ああ、暖かいね。町村会の方が真ん中に行かせてくれたからね」と意味不明。

の引用句中での句点は文の区切りではない。毎日新聞では、引用句は「 」を使って表現される (『』は「 」内で「 」と同じ働きをする)。

ここでは、引用句を「 」で囲まれた範囲として定義し、地の文の句点相当表現から句点相当表現の間を文と定義した。この定義に基づき、前処理をおこなった記事データ (段落単位になっている) を文に分割する。

毎日新聞 93 年 のある記事から不要部分を取り除いて、文ごとに分割する例を図 **2.2**, 図 **2.3** に示す。この記事では、段落ラベルや補充要素が排除されている。

◇高齢化社会をよくする女性の会講演会　１２日(火)午後１時半―４時、東京都千代田区神田駿河台３の９、三井海上火災保険本社ビル１階大会議室(ＪＲ御茶ノ水駅、地下鉄新御茶ノ水、淡路町、小川町駅下車)。スウェーデン研究で同国から北極星勲章を受けた岡沢憲芙早大教授が「ほんとうの『生活大国』スウェーデン事情」をテーマに講演する。問い合わせは同会(０３・３３５６・３５６４)へ。

図 **2.2.** 不要部分の削除例 (網掛は削除される部分であることを表わす)

１２日午後１時半―４時、東京都千代田区神田駿河台３の９、三井海上火災保険本社ビル１階大会議室。
スウェーデン研究で同国から北極星勲章を受けた岡沢憲芙早大教授が「ほんとうの『生活大国』スウェーデン事情」をテーマに講演する。
問い合わせは同会へ。

図 **2.3.** 文に分割された状態

2.4 形態素解析

形態素解析結果の例を図 **2.4** に示す。形態素解析には、意味付与ツールを用いる。出力フォーマットのオプションは以下のオプションを用いる。

(出力フォーマット "%m\t%Y\t%M\t%h/%t/%f\n")

解析結果は、表記/読み/原形/品詞番号の四つ組からなる。

```
１２　　イチニ　１２　　19/0/0
日　　{ジツ/ニチ}　　日　　33/0/0
午後　　ゴゴ　　午後　　16/0/0
１　　イチ　　１　　19/0/0
時半　　ジハン　時半　　33/0/0
―　　{―/カラ/タイ/ヒク}　　―　　73/0/0
４　　ヨン　　４　　19/0/0
時　　ジ　　時　　33/0/0
、　　、　　、　　75/0/0
東京　　トーキョー　　東京　　11/0/0
都　　ト　　都　　28/0/0
千代田　チヨダ　千代田　11/0/0
区　　ク　　区　　28/0/0
```

図 **2.4.** 形態素解析結果の例

2.5 形態素の正規化

一般の形態素解析システムでは、数字表現は全体で一つの形態素となる。そのままだと、アラビア数字と漢数字が同じ数を表現していても、別の語となってしまう。また、数を一つの形態素とすると、数の種類だけ語が増えてしまうため、無数に形態素の種類が増えることになる。

そこで、数字表現の正規化が必要となる。音声認識システムで用いることを考えると、読みが一意に決まる程度の単位に分割し、その細かい単位を組み合わせて数字表現を構成できる単位に正規化するのが望ましい。

新聞に現われる数字表現の分類を以下に示す。

1. 欧米式位取り表現 (1)　３６８，０００
2. 日本式位取り表現 (1)　３６万８０００
3. 日本式位取り表現 (2)　三十六万八千
4. 数字のみの表現　３６８０００, ３６・３３, ３６．３３
5. 欧米式位取り表現 (2)　３６８千
6. 時間　１０分３６秒２６

7. 数の併記　３・３・７拍子, １３・１４・１５日
8. 電話番号　０３・１１１１・２２２２

このうち、1 から 4 までは、ごく普通に位取りしながら読む。数字を位取りして読む場合には、「兆億万」の 4 ケタ単位でまず大きく区切られる。区切られた 4 ケタ内では、「|ゴセン|ハッピャク|キュウジュウ|ロク|」とケタごとに位取りして読む。

したがって、どの形式も、「|三十|六|万|八千|」という形に分割して表記を揃える。また、小数点以下に関しては、「|サンジュウ|ロク|テン|サン|サン|」と一桁ごとに分けて読むので、「|三十|六|・|三|三|」のような形に分割して表記を揃える。

時間の表現に関しては、単位ごとに区切り位取りしながら読む。ただし、秒以下に関しては、小数点以下の読み方と同様に区切って読む。

併記に関しては、区切り記号で区切ったあと、それぞれの部分を位取りしながら読めばよい。区切り記号は読まない。

電話番号に関しては、一桁ごとに区切って読む。区切り記号は、「ノ」と読んでもよいし、読まなくてもよい。

この処理をした結果を図 **2.5** に示す。

```
十      ジュー  十      19/0/0
二      ニ      二      19/0/0
日      ニチ    日      33/0/0
午後    ゴゴ    午後    16/0/0
一      イチ    一      19/0/0
時半    ジハン  時半    33/0/0
—       {—/カラ/タイ/ヒク}      —       73/0/0
四      ヨ      四      19/0/0
時      ジ      時      33/0/0
```

図 **2.5.** 読みを修正し数字を正規化した結果

2.6 出現頻度の計量

毎日新聞 1991 年 1 月から 1994 年 9 月分までの 45 か月分の記事に、これまでに述べたきた処理を行なった結果のデータ量を表 **2.2** に示す。

表 2.2. コーパスのデータ量

文数	2,371,932
段落数	1,438,311
記事数	281,818
形態素数	65,924,707
形態素 (種類)	220,928

この学習データに出現した形態素を形態素解析結果で得られた「表記/読み/原形/品詞番号」の四つ組が異なれば別の語であるとして出現頻度を計量した[1]。このとき高頻度語から順にある規模の語彙を選んだときに、データ全体のどれくらいの割合を占めるかを示す尺度を被覆率 (coverage) とよぶ。学習データに対する被覆率を表 2.3 に示す。

表 2.3. 被覆率

語彙規模	被覆率 (%)
5000	88.3
20000	96.4
24000	97.0
53000	99.0
101000	99.7
154000	99.9

2.7 高頻度語彙の構築

高頻度語 20000 語 (実際には、20036 語) を選んでシステムの語彙とする。このように選ぶため、「歩く」は語彙に含まれるのに「歩か (ない)」は語彙に含まれないという問題も生じる。

この高頻度語のリストから認識用の辞書を構築する。読みを併記したエントリは、認識用の辞書のエントリとしては別々のエントリとなる。読みを併記したエントリが展開されるため、認識用の辞書のエントリ数は、21380 となった。高頻度語に含まれる記号のうち、「%」のように読みがある記号は、通常の単語と同様に認識用辞書のエントリとするが、句

[1] これ以降の節では、CMU-Cambridge SLM toolkit[2] を使っている。この節の作業には、`text2wfreq` というコマンドを用いる。

```
日弁連+ニチベンレン+9          [日弁連]     n i ch i b e N r e N
日没+ニチボツ+2                [日没]       n i ch i b o ts u
日本+{ニホン/ニッポン}+12      [日本]       n i h o N
日本+{ニホン/ニッポン}+12      [日本]       n i q p o N
```

図 **2.6.** 音声認識用辞書のエントリの例

読点のように読みがない記号については、「、」「。」「？」のみを無音の音韻モデルに対応させ、それ以外の記号のエントリは削除した。

音声認識用辞書の例を図 **2.6** に示す。認識用辞書は、言語モデル用エントリ、認識結果用表記、音韻系列からなる。なお、言語モデル用エントリは前述の四つ組みであるが、表記と原形が同じ場合には省略し、各要素を+で結合した形であらわしている。音韻系列は読みを認識システムの音素体系に変換したものである。また品詞番号のうち冗長な 0 は削除する。

2.8 単語 N-gram の計量

形態素解析済みの学習データから単語の N 組のリストを計量する。このとき、文頭と文末を表わす <s> と </s> を文データに付加する。ここでは、バイグラムと逆向きトライグラムを作成するので、それぞれ計量する。逆向きトライグラムの例を図 **2.7** に示す。この例では、たとえば、「米国」のあとに「と」が出現して、そのあとに「ＥＣ」が出現した回数が、全部で 104 回あったことを示している。

```
ＥＣ+イーシー+9 と+ト+64 米国+ベイコク+12 104
ＰＫＯ+ピーケーオー+4 の+ノ+67 日本+{ニホン/ニッポン}+12 138
ＰＬＯ+ピーエルオー+9 と+ト+64 イスラエル+イスラエル+12 103
あげ+アゲ+あげる+44/6/5 を+ヲ+58 成果+セイカ+2 143
あげ+アゲ+あげる+44/6/5 を+ヲ+58 声+コエ+2 116
あこがれ+アコガレ+2 の+ノ+67 へ+エ+58 108
あっ+アッ+ある+44/17/8 が+ガ+58 こと+コト+21 902
```

図 **2.7.** 逆向きトライグラムの例

学習データ中に出現した単語の三つ組、二つ組の種類を表 **2.4** に示す。

表 **2.4.** N-gram の種類数

トライグラム	18,393,056
バイグラム	4,074,644

2.9 制限語彙 N-gram の構築

単語 N-gram にあらわれる高頻度語以外の語は全て一つの語 (ここでは、<UNK> と表わす) として N-gram をまとめる。たとえば、

こと+コト+21 な+ナ+だ+70/48/7 愚か+オロカ+18

という三つ組に含まれる「愚か」が高頻度語でない場合は、この三つ組は

こと+コト+21 な+ナ+だ+70/48/7 <UNK>

という三つ組として、他の同様な三つ組とまとめられる。制限語彙の N-gram の種類数を表 **2.5** に示す。

表 **2.5.** 制限語彙 N-gram の種類数

トライグラム	14,465,830
バイグラム	2,322,666

2.10 バックオフ N-gram モデルの構築

制限語彙 N-gram をもとに、バックオフ N-gram モデルを構築した。バイグラムのカットオフを 1, 4、トライグラムのカットオフを 2, 4、として作成したモデルのエントリ数を表 **2.6** に示す。

表 **2.6.** バックオフ N-gram モデルのサイズ

カットオフ	1 1	4 4
トライグラム	4,733,916	1,593,020
バイグラム	1,238,929	657,759

参考文献

[1] 荻野紫穂. リストのラベルとして使われる丸括弧とリストの範囲. 計量国語学, Vol. 19, No. 4, 1994.

[2] Ronald Rosenfeld. The CMU Statistical Language Modeling Toolkit and its use in the 1994 ARPA CSR Evaluation. In *Proc. ARPA Spoken Language Systems Technology Workshop*, pp. 47–50, January 1995.

付 録A　高頻度語 6 万語言語モデル†

A.1 言語モデル及び、音声認識辞書の作成手順

本報告書に含まれる高頻度語 6 万語による言語モデル、音声認識辞書の構築手順について説明する。学習用コーパスとして毎日新聞 91 年 1 月から 94 年 9 月、および 95 年 1 月から 97 年 6 月までの 75ヶ月文を使用した。　コーパスからは、発話が困難な表現は除去してある。コーパスのデータ量は 4,290,921 文、117,800,320 形態素、形態素の種類は 253,502 である。

A.1.1 学習用テキストの作成

日本語の表記は分かち書きされていないため、形態素解析などを用いて文章を N-gram の単位に分割する必要がある。形態素解析システム ChaSen 2.0b6 を使用して文章を分割し、後処理としては、

- `ChaWan 2.03` により、数詞・助数詞等の読み修正。
- `suuzi_syori` により、数字表現の正規化としての数字の位取り。
- `postprocess 1.0` により、語幹の読み変化などの読みの修正。

の処理を行った。N-gram 構築の単位は、”形態素” + ”読み” + ”原形” + ”品詞”を用いた。

A.1.2 N-gram 言語モデルの構築

N-gram 言語モデルは、CMU SLM Toolkit を使用して構築した。語彙は学習コーパスにおける高頻度語 6 万語を使用し、bigram、および逆向き trigram を構築した。6 万語での単語被覆率は 99.22% である。カットオフは bigram,trigram 共に 1 とし、は Witten-Bell ディスカウンティングを用いて構築を行った。

† 伊藤 克亘 (電子技術総合研究所 知能情報部)

N-gram パラメータ数は、bigram $2,420,231$, trigram $8,368,507$ である。また、N-gram パラメータ削減プログラムにより、trigram パラメータを 10% に圧縮し、$836,852$ とした言語モデルを構築した。

A.1.3 音声認識辞書の作成

音声認識辞書は、CMU SLM Toolkit の語彙ファイルから `vocab2htkdic.pl` を使用して作成した。辞書は HTK の dictionary format とほぼ同じフォーマットである。

辞書作成時に形態素解析システムにおいて未知語と出力された形態素は、基本的に削除した (形態素解析誤りなどの原因による)。しかし、形態素がアルファベットである場合は、人手により読みを付与する事によって辞書を修正した。読みを併記したエントリは、認識用の辞書エントリとしては別々のエントリとなるため、音声認識辞書のエントリ数は、$63,534$ となった。

付 録B　CMU-Cambridge 統計的言語モデルツールキット†

B.1 はじめに

CMU-Cambridge Statistical Language Modeling Toolkit (CMU-ケンブリッジ統計的言語モデルツールキット) は，カーネギーメロン大学の Ronald Rosenfeld と，ケンブリッジ大学の Phillip Clarkson によって書かれた，N-gram 言語モデル作成のためのプログラム群である．このツールキットは，

- 多くのコマンド群からなる
- 容易に N-gram が構築できる
- 高速な処理，圧縮ファイルのサポート
- 任意の N についての N-gram モデルのサポート
- 4 種類の back-off

という特徴を持つ．このツールキットを使うことで，以下のような作業を簡単に行なうことができる．

- 単語頻度リストの作成
- 語彙リストの作成
- back-off N-gram 言語モデルの作成
- back-off N-gram 言語モデルの評価

また，このツールキットは，研究目的であれば無料で利用・配布することができる．

† 伊藤 彰則 (山形大学 工学部)、田本 真詞 (NTT 基礎研究所)

B.2 ファイル形式

CMU-Cambridge ツールキットでは，以下のようなファイルを扱う．

- .text

  統計の元となる文であり，単語が空白 (半角) で区切られていることを仮定している．例えば

  > 我輩 は 猫 であ る 。
  > 名前 は まだ な い 。

  のような形式である．文脈情報 (コンテキストキュー) として，<s>(文頭)，</s>(文末)，<p>(パラグラフ) のような特殊記号を使うこともできる．これらの記号は，自分で定義することが可能である (後述の.ccs ファイル参照)．

- .wfreq

  単語とその出現頻度の組である．1 行に 1 つの単語の情報が入る．例えば次のようなものである．

  > アドバンスト 1
  > 衛星通信 17
  > 相談所 9
  > 現象 390
  > 現職 97
  > 現場 569
  > ⋮

- .vocab

  語彙リスト．1 行に 1 つの単語が記述され，単語の文字コード順にソートされる．行の先頭が ##の行はコメントとみなされる．例えば次のようなものである．

  ```
  ## Vocab generated by v2 of the CMU-Cambridge Statistcal
  ## Language Modeling toolkit.
  ##
  ## Includes 20000 words ##
  ```

  、

。
、
・
・
：
？
！
〃
々
〇

- `.idngram (id2gram, id3gram, ...)`

  N-gram count のファイル．通常はバイナリ形式だが，テキスト形式で出力することもできる．各単語は単語番号に置きかえられ，未知語は 0 に置きかえられる．

- `.wngram (w2gram, w3gram, ...)`

  N-gram count のファイル (テキスト)．単語の組の文字コード順に並んでいる．通常の N-gram 作成では使わない．

- `.ccs`

  コンテキストキュー (<s> や<p> など) を記述する．ここで指定された記号は，コンテキストとしては使われるが，言語モデルの予測には使われなくなる．

- `.binlm`

  作成された言語モデル (バイナリ形式)．

- `.arpa`

  作成された言語モデル (ARPA 形式，テキストファイル)．

B.3 言語モデルの作成の作成と評価

B.3.1 言語モデルの作成

言語モデル作成の一般的な手順を図 B.1 に示す．これは，次のような手順からなる．

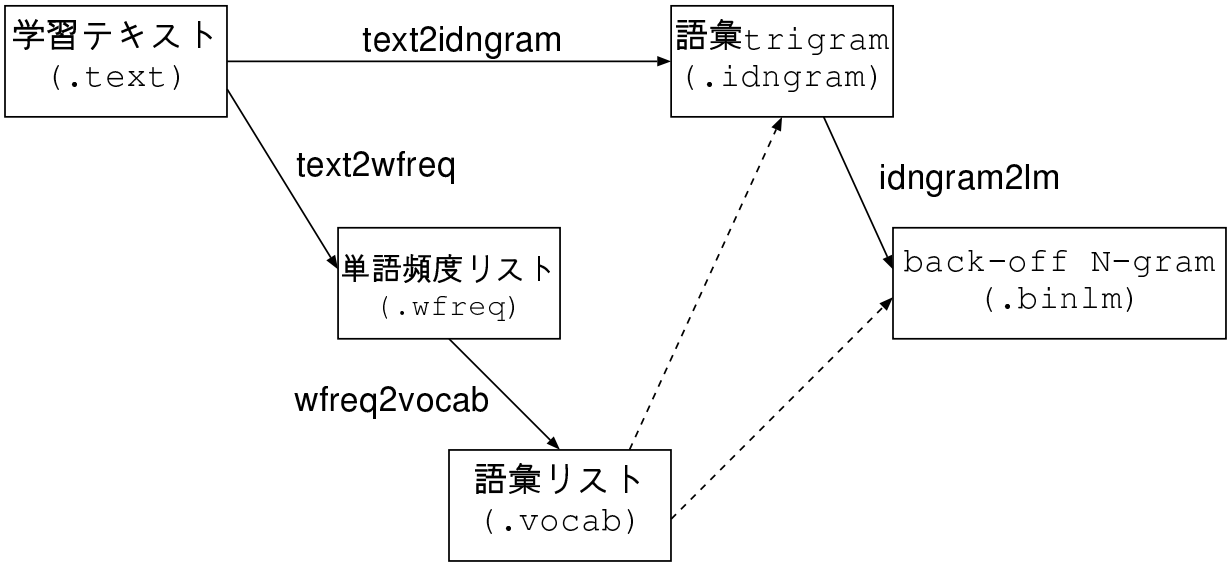

図 **B.1.** 言語モデル作成手順

1. 学習テキストから単語頻度リストを作る．
   これには，text2wfreq コマンドを使う．例えば，

   ```
   text2wfreq < learn.text > learn.wfreq
   ```

   のようにする．学習テキスト量が多い場合には，ファイルが圧縮されている場合が多い．gzip で圧縮してある場合には

   ```
   gzip -dc learn.text.gz | text2wfreq | gzip > learn.wfreq.gz
   ```

   また，compress で圧縮してある場合は

   ```
   zcat learn.text.Z | text2wfreq | compress > learn.wfreq.Z
   ```

   のようにすればよい．
2. 単語頻度リストから語彙リストを作る．
   単語頻度リストの上位 n 個，または頻度 m 回以上の単語を語彙とする．これには，wfreq2vocab コマンドを使う．例えば頻度の上位 5000 個を語彙とする場合には，

   ```
   wfreq2vocab -top 5000 < learn.wfreq > learn.vocab5k
   ```

   のようにする．結果のファイルの拡張子 (.vocab5k) は，5000 語彙のリストという意味だが，特にこういう形式でなければならないということはなく，適当に付けてもよい．
   上位 n 個という指定ではなく，例えば 30 回以上出現した単語を語彙とする場合には，

```
wfreq2vocab -gt 29 < learn.wfreq > learn.vocab5k
```

とする (gt は grater than の略)．

3. 学習テキストと語彙リストから ID n-gram を作る．
   語彙が決まったら，ID n-gram を作る．これには，

```
text2idngram -vocab learn.vocab5k < learn.text > learn.id3gram
```

   のようにする．text2wfreq の場合と同じように，圧縮ファイルを扱うこともできる．
   text2idngram には，次のようなオプションがある．

   -buffer バッファサイズ (MB)
   : 入力のソートに使うメモリサイズを指定する．デフォルトの値はコンパイル時に指定するが，そのままコンパイルすると 100MB に設定される．

   -temp ディレクトリ
   : ソートのための一時ファイルを置くディレクトリを指定する．

   -files ファイル数
   : 一時ファイルのために，一度に open するファイルの数を指定する．デフォルトは 25．

   -n 長さ
   : 生成される n-gram の長さを指定する．デフォルトは 3．

   -write_ascii
   : バイナリでなく，テキスト形式で出力する．

   -fof_size サイズ
   : text2idngram は，ある出現頻度の単語が何種類あるか (Frequency-of-Frequency,fof) を出力する．このオプションは，fof のリストのサイズを指定する．デフォルトは 25．

   -verbosity 数字
   : 実行時に，どの程度の情報を出力するかを制御する．0 を指定すると，余計な情報の出力をしない．デフォルトは．2

4. ID n-gram と語彙リストから back-off 言語モデルを作る．
   最終的に，back-off 言語モデルを作成する．これには，idngram2lm コマンドを使う．

```
idngram2lm -idngram learn.id3gram -vocab learn.vocab5k -binary
learn.binlm
```

これで，back-off 言語モデルのファイル learn.binlm が生成される．idngram2lm は，圧縮ファイルにも対応しており，

```
idngram2lm -idngram learn.id3gram.gz -vocab learn.vocab5k.gz
-binary learn.binlm
```

のように，入力として圧縮したファイルを指定することもできる．
出力として，バイナリ形式の binlm でなく，テキスト形式 (.arpa) を選ぶ場合は，

```
idngram2lm -idngram learn.id3gram -vocab learn.vocab5k -arpa learn.arpa
```

のように指定する．
idngram2lm には，次のようなオプションがある．

-context コンテキストキュー
: コンテキストキューファイル (.ccs) を指定する．

-calc_mem | -buffer バッファサイズ | -spec_num n-gram 数 n-gram 数...
: これらのオプションは，言語モデル作成の際のメモリ使用に関するオプションである．-calc_mem を指定すると，ID n-gram のファイルを 1 回読んで，各 n-gram に割りあてるメモリ量を計算した後，もう一度 ID n-gram ファイルを読みこむ．このオプションを使うと，時間がかかるかわりに，メモリの利用を最適化することができる．-buffer は，言語モデル作成のために確保するメモリ量を MByte 単位で指定する．デフォルトは 100 である．-spec_num は，2-gram, 3-gram, ... の個数を指定する．このオプションを指定すると，その指定にしたがってメモリ利用を最適化する．ここに指定すべきパラメータは，text2idngram を実行したときに表示される．

-vocab_type 0 ~ 2
: 未知語を含むモデル (open vocabulary model) の場合は 1，未知語を含まないモデル (closed vocabulary model) の場合は 0 を指定する．学習データに未知語を含まないが，入力には未知語を許すという場合には 2 を指定する．デフォルトは 1．

-oov_fraction 比率
: 上記の-vocab_type オプションで 2 を指定した場合の，未知語の比率を指定する．ディスカウントされた確率のうち，ここで指定した比率が未知語に回される．デフォルトは 0.5．

-linear | -absolute | -good_turing | -witten_bell

ティスカウントの手法を指定する．各オプションに対応するディスカウント手法は次の通り．

`-linear`	linear discounting
`-absolute`	absolute discounting
`-good_turing`	Good-Turing discounting
`-witten_bell`	Witten-Bell discounting

デフォルトは Good-Turing discounting である．

`-disc_ranges` 回数 1 回数 2 回数 3 ...

Good-Turing discounting を用いる場合，各 n-gram でどこまで discounting をするかを指定する．デフォルトは，unigram が 1，bigram が 7，trigram が 7 である．

`-cutoffs` 回数 2 回数 3 ...

各 n-gram のカットオフを指定する．「回数 2」が bigram のカットオフ，「回数 3」が trigram のカットオフである．ここで指定した回数以下の n-gram count は削除される．デフォルトはすべて 0 である．

`-min_unicount` 回数

unigram の出現回数の最小値を指定する．ここで指定した回数よりも出現頻度の少ない unigram は，強制的にこの回数だけ出現したことにして集計される．

`-zeroton_fraction` ζ

未知語の確率 $P(\text{zeroton})$ を，1 回だけ出現した単語の確率 $P(\text{singleton})$ の ζ 倍に設定する．デフォルトは 1.0．

`-ascii_input` | `-bin_input`

入力の ID n-gram がテキスト形式のときは`-ascii_input`，バイナリ形式のときは`-bin_input` を指定する．

`-n` 長さ

n-gram の長さを指定する．デフォルトは 3．

`-verbosity` 数字

実行時に，どの程度の情報を出力するかを制御する．0 を指定すると，余計な情報の出力をしない．デフォルトは．2

`-four_byte_counts`

デフォルトでは，n-gram の「出現回数の種類」(出現回数そのものではない) が

65535 種類までしか扱えない．これが 65535 回を越える場合，このオプションを指定する．

-two_byte_bo_weights

Back-off の重みを 2 バイト整数で貯える．これにより，メモリ消費量が少なくなるが，そのかわり誤差が増大する．

B.3.2 言語モデルの評価

作成した言語モデルは，単語 perplexity により評価する．モデルの評価をするコマンドが evallm である．evallm は，対話的に処理を行なうプログラムである．例えば次のような使いかたをする．(下線部がユーザーの入力)

```
% evallm -binary learn.binlm
Reading in language model from file learn.binlm
Done.
evallm :  perplexity -text test.text
Computing perplexity of the language model with respect
  to the text jconstp.text
Perplexity = 32.25, Entropy = 5.01 bits
Computation based on 232 words.
Number of 3-grams hit = 136 (58.62%)
Number of 2-grams hit = 59 (25.43%)
Number of 1-grams hit = 37 (15.95%)
135 OOVs (36.78%) and 0 context cues were removed from the
calculation.
evallm :  quit
evallm :  Done.
```

基本的な使い方としては，

evallm -binlm 言語モデルのファイル（バイナリ）

または

```
evallm -arpa 言語モデルのファイル（テキスト）
```

で evallm を起動し，evallm: のプロンプトで

```
perplexity -text 評価テキスト
```

を入力する．この場合の評価テキストは，学習テキストと同じく，単語間を空白で区切ったテキストファイルでなければならない．

evallm の起動時のオプションとして，

```
-ccs コンテキストキュー
```

で，コンテキストキューのファイルを指定することができる．また，起動した後のコマンドとして，次のものが使える．

perplexity -text 評価テキスト
: 指定されたファイルのテストセットパープレキシティを計算する．このコマンドには，次のようなオプションがある．

  -probs ファイル
  : 評価テキストの各単語の出現確率を，指定したファイルに書き出す．

  -oovs ファイル
  : 評価テキストに出現した未知語を，指定したファイルに書き出す．

  -annotate ファイル
  : 評価テキストの各単語の確率，対数確率，計算状況 (直接求まったか，back-off したか，etc.) を，指定したファイルに書き出す．

  -backoff_from_unk_inc | -backoff_from_unk_exc
  : コンテキストに未知語が含まれる場合には，強制的に back-off する．
  -backoff_from_unk_inc を指定した場合，

$$P(w|w', !!\text{UNK}!!) = P(w|!!\text{UNK}!!)$$

  として計算し，-backoff_from_unk_exc を指定した場合，

$$P(w|w', !!\text{UNK}!!) = P(w)$$

  として計算する．

-backoff_from_ccs_inc | -backoff_from_ccs_exc

コンテキストにキュー (.ccs ファイルで指定する内容．<s>など) が含まれる場合は，強制的に back-off する．inc と exc の違いは，上と同じ．

`-backoff_from_list` ファイル

「その単語がコンテキストに来たら強制的に back-off する単語」のリスト (force back-off list) を指定する．

-include_unks

未知語の出現確率を含めて perplexity を計算する．

`validate` w_1 w_2 ...

与えられたコンテキスト $w_1 w_2 \ldots$(trigram の場合は 2 単語) において，

$$\sum_w P(w|w_1 w_2 \ldots) = 1$$

になるかどうかをチェックする．オプションとして`-backoff_from_unk_inc`, `-backoff_from_unk_exc`, `-backoff_from_ccs_inc`, `-backoff_from_ccs_exc`, `-backoff_from_list` が使える．

`help`

コマンド一覧を表示する．

`quit`

`evallm` を終了する．

B.3.3 その他のコマンド

以上，ツールキットの基本的な使い方と，最もよく使うコマンドについて解説した．CMU-Cambridge ツールキットには，これ以外にも多くのコマンド群が含まれる．ここでは，それらのコマンドについて簡単に説明する．詳しい解説は，ツールキットに含まれるドキュメントを参照のこと．

`text2wngram`

テキストを，n-gram(単語番号に変換されていないもの) にする．次のようなオプションがある．

`-n` 長さ
`-temp` ディレクトリ
`-chars` 一度に処理する最大文字数

-words 一度に処理する最大単語数
-verbosity 数字

ngram2mgram -n 数字 -m 数字
n-gram の長さを変換する．オプションは次の通り．

-ascii -binary -words

wngram2idngram
単語 n-gram を ID n-gram に変換する．オプションは以下の通り．

-vocab 語彙ファイル
-buffer バッファサイズ
-hash ハッシュ表のサイズ
-temp ディレクトリ
-files 同時に開くファイル数
-verbosity 数字
-n 長さ
-write_ascii

idngram2stats
ID n-gram から，その frequency-of-frequency の値，各 n-gram の個数などの統計を表示する．

-n 長さ
-fof_size freq-of-freq リストのサイズ
-verbosity 数字
-ascii_input

mergeidngram $idngam_1$ $idngram_2$ ...
複数の ID n-gram を合わせて 1 つの ID n-gram を生成する．

-n 長さ
-ascii_input
-ascii_output

binlm2arpa -binary xxx.binlm -arpa zzz.arpa
バイナリ形式の言語モデルを，テキスト形式に変換する．

`interpolate` +確率ファイル 1 +確率ファイル 2 ...

複数の確率ファイル (`evallm` の `perplexity -probs` で出力されるもの) を線形補間した場合の，最適なパラメータを計算する．オプションは次の通り．

-test_all
-test_first 数
-test_last 数
-cv
-tag ファイル
-captions ファイル
-in_lambda 初期値ファイル
-out_lambda ファイル
-probs ファイル
-max_probs 個数

B.4 プログラムからの toolkit の利用

CMU-Cambridge toolkit として公開されているのは，個々のコマンドとその利用法のみである．しかし，実際にいろいろな実験をする上で，自作のプログラムの中から言語モデルの評価をしたりする必要もあるであろう．そのための API は公開されてはいないが，CMU-Cambridge toolkit はソースで配布されているので，内部の関数の利用法は比較的簡単に調べることができる．

ここでは，バイナリ形式の言語モデル (binlm) から確率を計算する方法を簡単に説明する．

B.4.1 必要なファイル

CMU-Cambridge toolkit のアーカイブを展開してできるディレクトリの下の，`lib/SLM2.a` が必要なライブラリである．また，`src/` の下のいくつかのインクルードファイルを利用する．

B.4.2 使用例

まず，使用例を示す．この例は，標準入力から 3 つの単語を入力し，その確率を表示する．言語モデルは `model.binlm` というファイルから読み，trigram を仮定している．

```
#include <stdio.h>
#include <SLM2.h>

main()
{
    ng_t model;
    char w1[40],w2[40],w3[40];
    int bo_case;
    int n[3],i;
    id__t w[3];
    double prob;

    /* 言語モデルの読みこみ */
    load_lm(&model,"Genesis.binlm");

    /* 単語の入力 */
    printf("Input three words:");
    fflush(stdout);
    scanf("%s%s%s",w1,w2,w3);

    /* 単語の文字列を ID に変換 */
    sih_lookup(model.vocab_ht,w1,&n[0]);
    sih_lookup(model.vocab_ht,w2,&n[1]);
    sih_lookup(model.vocab_ht,w3,&n[2]);

    /* 確率の計算 */
    for (i = 0; i < 3; i++)
        w[i] = n[i];
    bo_ng_prob(2,w,&model,2,&prob,&bo_case);
    printf("%f\n",prob);
}
```

CMU-Cambridge toolkit が /usr/local/cmutk 以下にインストールされていたとすると，このプログラムは以下のコマンドでコンパイルできる．

```
cc -I/usr/local/cmutk/include sample.c
cc -o sample sample.o /usr/local/cmutk/lib/SLM.a -lm
```

ただし，このときの C コンパイラは ANSI 準拠でなければならない．

B.4.3 各関数の解説

次に，この例に使われている関数の簡単な説明をしてみよう．

```
void load_lm(ng_t *ng,char *lm_filename)
```

言語モデル (バイナリ形式) を読みこむ．

```
void sih_lookup(sih_t *ht, char *string, int32 *intval)
```

ハッシュ表 ht を検索し，文字列 string に対応する整数を探す．結果は *intval に格納され，文字列が見つからなかった場合は 0 が格納される．実際の利用では，

```
ng_t model;
char *word;
int id_no;
sih_lookup(&model.vocab_ht,word,&id_no);
```

のように利用する．

```
void bo_ng_prob(int context_length,
                id__t *sought_ngram,
                ng_t *ng,
                int verbosity,
                double *p_prob,
                int *bo_case)
```

n-gram 確率を計算する．context_length は，コンテキストの長さ (n-gram の長さ -1)．sought_ngram は，単語番号が格納された配列である．ここで，単語番号の型である id__t は実質的に unsigned short なので，前述の sih_lookup() の引数に直接 id__t の変数を書くことはできない．ng は言語モデルへのポインタ，verbosity は処理内容の出力の程度を示す．計算結果は*prob に格納され，どのように計算されたかの情報が*bo_case に格納される．

付録：CMU-Cambridge toolkit のインストール

CMU-Cambridge toolkit のインストールは，version 1 に比べて非常に楽になった．まず，

```
http://svr-www.eng.cam.ac.uk/~prc14/toolkit.html
```

から，ソースアーカイブ CMU-Cam_Toolkit_v2.tar.gz をダウンロードして，展開する．

```
gzip -dc CMU-Cam_Toolkit_v2.tar.gz | tar xvf -
```

すると，CMU-Cam_Toolkit_v2 というディレクトリが作成され，その下に必要なファイルが展開される．

次に，CMU-Cam_Toolkit_v2/src の下の Makefile を編集する．変更の必要があるのは，37 行目の

```
#BYTESWAP_FLAG    = -DSLM_SWAP_BYTES
```

の行である．little-endian のコンピュータ[1](VAX, i386/486/Pentium など) でコンパイルする場合には，この行の先頭の #を削除する．自分のコンピュータが little-endian かどうかわからない場合には，CMU-Cam_Toolkit_v2 の下にある endian.sh というスクリプトを動かしてみると，little-endian かどうかを教えてくれる．

Makefile を修正したら，そのディレクトリで

```
make install
```

を実行すると，CMU-Cam_Toolkit_v2/bin の下に各コマンドの実行ファイルができ，また CMU-Cam_Toolkit_v2/lib の下に SLM2.a というライブラリができる．あとは，CMU-Cam_Toolkit_v2/bin を shell の PATH に追加すれば，インストールは完了である．

CMU-Cambridge toolkit に関する詳しいドキュメントは，CMU-Cam_Toolkit_v2/doc の下の toolkit_documentation.html にある．HTML 形式なので，適当な WWW ブラウザで読むとよい．

[1] 例えば，0x1234 という 2 バイト整数をファイルに書き出したときに，下位のバイトから 0x34 0x12 という並びになるのが little-endian のコンピュータである．逆に，上位のバイトから 0x12 0x34 という並びになるのが big-endian である．8086 ~ Pentium や VAX などは little-endian で，Sparc や m68k などは big-endian である．中には，MIPS のように little-endian と big-endian を切りかえられる CPU もある．

付 録C　N-gram パラメータ削減プログラムのアルゴリズム概要†

C.1 エントロピーに基づく逐次削減手法

この手法は、パラメータ削除による言語モデルの性能低下を、エントロピー尺度に基づいて定量的に、パラメータ毎に評価するものであり [1-3]、back-off 係数を更新していく点、任意のパラメータ数を実現できるという点に特徴がある。ここでは、簡単化のために trigram を例にとって説明することにする (図 **C.1**)。プロセスは、次のようになる。

1. back-off N-gram モデルを作成する (ディスカウンティング手法は任意)。
2. あるコンテキスト、すなわち N-1 単語列を固定して考えた時、次の二つの確率分布を得る

   $\{p\}$　元のモデルの条件付き確率分布

   $\{p'\}$　一つの N-gram パラメータを削除し、back-off により推定する場合に得られる確率分布
3. 以上の二つの分布と、新しい back-off 係数 (α') を用いて、エントロピーの増加量を求める。その値は二つの分布間の相対エントロピーにコンテキストの頻度をかけることによって求められる。
4. エントロピーの増加量が小さいものから、あるいは与えられたしきい値以下のものを削除し、必要であれば back-off 係数を更新する (終了)。

この過程における、back-off 係数の更新方法および、エントロピーの増加量の算出法について整理する。

† 踊堂 憲道 (奈良先端科学技術大学院大学 情報科学研究科)

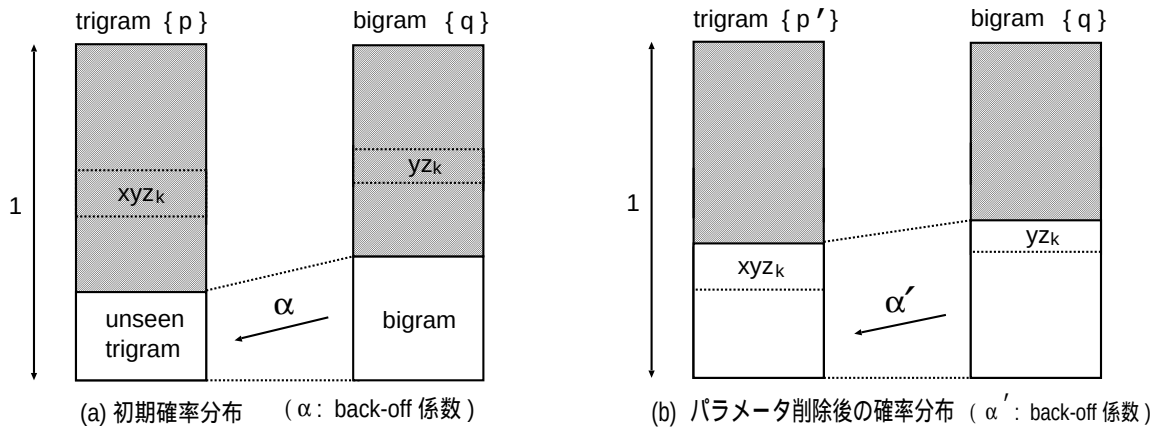

図 **C.1.** back-off 係数の更新

C.1.1 back-off 係数の更新法

trigram モデルを考える場合、コンテキストである 2 単語列によってモデルの空間、すなわち全ての trigram パラメータを分類することができる。ここでは、3 単語列 $xyz_1, xyz_2, \ldots$ は、「部分空間 Q_{xy} に含まれる」という表現を用いて、「各部分空間は他の部分空間に対して独立である」と仮定する。今、部分空間 Q_{xy} だけを考えて、削除前の trigram パラメータ $xyz_i(i = 1, 2, \ldots)$ の確率分布を $\{p\}$、対応する bigram パラメータ yz_i の確率分布を $\{q\}$ とする。また、未知の trigram および bigram に与えられる確率値の総和をそれぞれ p_{unks}、q_{unks}[1] とする。このとき、back-off 係数 α は、$\boldsymbol{c}_+$ を $C(xyz_i) > 0$ である i の集合とすると、

$$\alpha = \frac{p_{unks}}{q_{unks}} = \frac{1 - \sum_{i \in \boldsymbol{c}_+} p_i}{1 - \sum_{i \in \boldsymbol{c}_+} q_i} \tag{C.1}$$

となる (図 **C.1**(a))。今、trigram パラメータ xyz_k を削除する場合、$p_k = P(z_k|xy)$ を bigram から back-off smoothing により推定することになる。

このとき、新しい back-off 係数を以下の値に更新する (図 **C.1**(b))。

$$\alpha' = \frac{p'_{unks}}{q'_{unks}} = \frac{p_{unks} + p_k}{q_{unks} + q_k} \tag{C.2}$$

最終的に、初期状態の確率分布 $\{p\}$ と、(一つの) パラメータ削除後の分布 $\{p'\}$

$$\{p'\} = \begin{cases} p_i & i \neq k, i \in \boldsymbol{c}_+ \\ \alpha' \cdot q_k & k \in \boldsymbol{c}_+ \\ \alpha' \cdot q_i & i \notin \boldsymbol{c}_+ \end{cases} \tag{C.3}$$

を得ることになる。

[1] "unks" は unknown words を意味する

C.1.2 エントロピー変化量

一般に、「言語モデルが表現する言語の複雑さ」を表わす尺度として、エントロピーが用いられる。3 単語列 xyz の生起確率を $p(xyz)$ で表すと、エントロピーは、

$$\mathcal{H} = -\sum_{xyz} p(xyz) \log p(xyz) \tag{C.4}$$

で表され、最尤推定式と等価である。部分空間 Q_{xy} の生起確率を $P(Q_{xy})$ とすると、式 (**C.4**) は、trigram 確率 (条件付き確率) $p_i = P(z_i|xy)$ を用いて、

$$\begin{aligned} &= -\sum_{xy} P(Q_{xy}) \log P(Q_{xy}) \\ &\quad + \sum_{xy} P(Q_{xy})(-\sum_{i} p_i \log p_i) \end{aligned}$$

と書き直すことができる。

さて、全空間でのエントロピーの変化量は、部分空間 Q_{xy} における変化量と等しいと仮定しているので、$C(\cdot)$ を事象の生起回数として、

$$\begin{aligned} \Delta\mathcal{H} &= \mathcal{H}' - \mathcal{H} \\ &= P(Q_{xy}) \sum_{i} p_i \log \frac{p_i}{p'_i} \\ &= \frac{C(xy)}{C(all)} \times \mathcal{D}(p||p') \end{aligned} \tag{C.5}$$

となる。ここで、$C(all)$ は、bigram の総出現回数であり、$\mathcal{D}$ は、相対エントロピーである。

同様に、パープレキシティーの変化量は、空間 Q_{xy} だけで考えると、

$$\begin{aligned} \log(\Delta\mathcal{P}\mathcal{P}) &\propto \sum_{i} C(xyz_i) \log \frac{p_i}{p'_i} \\ &= C(xy) \times \mathcal{D}(p||p') \end{aligned} \tag{C.6}$$

という式で求めることができる。

参考文献

[1] 踊堂他: 情報量に基づく trigram パラメータの逐次的削減手法, 情報処理学会研究報告, SLP22-17, pp.91-96 (1998).

[2] N.Yodo et al. : Compression Algorithm of Trigram Language Models Based on Maximum Likelihood Estimation, Proc.ICSLP-98,pp.716-719 (1998).

[3] A.Stolcke: Entropy-based pruning of back-off language models, Proc.Broadcast News Transcription and Understanding Workshop, pp.27 0-274 (1998).

付 録D　N-gram パラメータ削減プログラム使用説明書†

D.1 はじめに

本プログラムは、CMU ならびに Cambridge で開発された統計的言語モデル作成用ツール CMU-Cambridge Toolkit Version 2.1 (以下 Toolkit) をベースに開発されたものであり、Toolkit で生成される back-off N-gram に対してパラメータ削減を行なう。

プログラムで扱うデータ構造は、Toolkit に準拠しており、入力は、Toolkit の binary 形式モデル、出力は同じく ARPA 形式ファイルである。また、本プログラムは、bigram (N=2) および trigram (N=3) にのみ対応している。

D.2 インストール方法

1. まず、Toolkit がインストールされているか、アーカイブ SLM2.a が作成されているかを確認する (Toolkit は以下のサイトから Download 可能)。
   http://svr-www.eng.cam.ac.uk/~prc14/toolkit.html

2. 適当なディレクトリで、compress.tgz を展開する。% gtar -zcvf compress.tgz
   compress.tgz は、以下のファイルを含んでいる。

   a. 削減プログラム
      LMcompress.c　compress2gram.c　compress3gram.c

   b. アーカイブ用のソース
      write_lms_2.c

   c. readme.j　ファイル構成、インストール手順に関する記述
      readme.e (英語版)

† 踊堂 憲道 (奈良先端科学技術大学院大学 情報科学研究科)

d. Tutorial.j　コマンドの使用例などに関する説明
Tutorial.e (英語版)

e. Makefile

3. コンパイル
Makefile の CMU_DIR に、 Toolkit のディレクトリを指定して、make を実行する。Toolkit で提供されるアーカイブ SLM2.a に write_lms_2.o が追加され、現在のディレクトリに、
アーカイブ Compress.a
実行可能ファイル LMcompress
が作成される。

4. インストール
Makefile の BIN 変数で指定されたディレクトリに、実行可能ファイル LMcompress をコピーする。
% make install

4. オブジェクト・ファイルの削除
% make clean

Makefile

Makefile の中の主な変数を以下に挙げておく。適宜変更して使用されたい (コンパイラは gcc などの ANSI C 準拠ものを指定してする等)。

BIN	インストール先ディレクトリ
BYTESWAP_FLAG	バイトスワップ・フラグ
CC	コンパイラ (default: gcc)
CMU_DIR	Toolkit がインストールされているディレクトリ
GDB_FLAG	デバッガ
OPT_FLAG	最適化フラグ (default: -O)
WARNING_FLAG	警告メッセージ

D.3 オプション説明

-a output filename ··· 出力ファイル (arpa, .gz ファイル対応)

-b input filename ··· 入力ファイル (binary, .gz ファイル対応)

-h ヘルプメッセージ出力

-p value ··· 削除後のパラメータ数の割合を指定 (%) (0.0 – 100.0)

コマンドの具体的な使用法については、Tutorial を参照されたい。

付 録E　単語正解率判定ツール説明書†

単語正解率判定ツールは、正解単語列と仮説単語列のアラインメントを行うプログラム、スコアリングを行うプログラムから構成されている。また、Julius を用いた大語彙連続音声認識結果のスコアリングを行うという本ツールの主目的に従って、Julius の認識結果ログから仮説単語列ファイルを作成するプログラムも含んでいる。ここでは、ツールの概要やこれらのプログラムの使用法等について説明を行う。

E.1 データフロー

本ツールのデータフローを図 E.1 に示す。

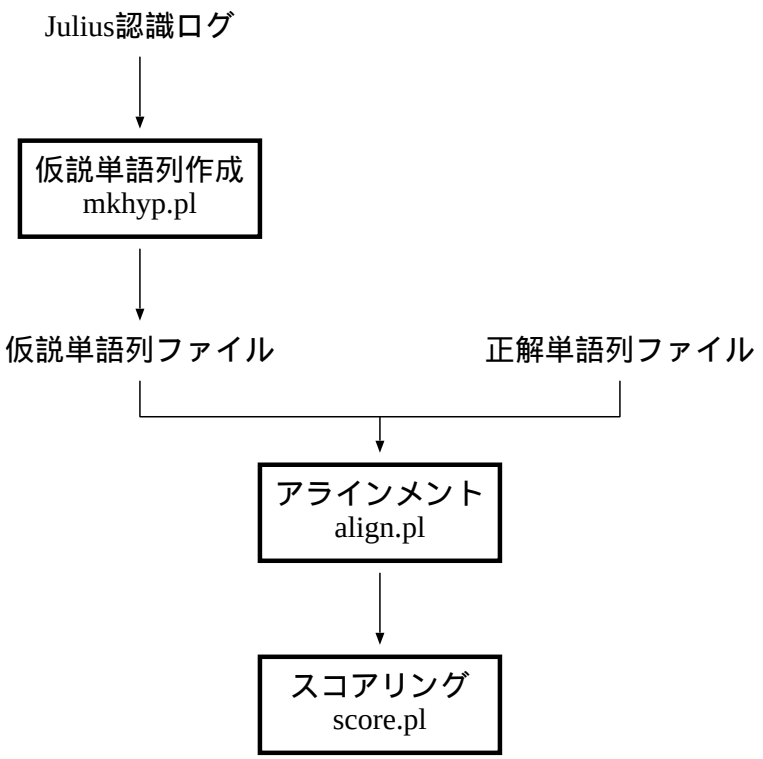

図 E.1. 単語正解率判定ツールのデータフロー

† 山本 俊一郎 (奈良先端科学技術大学院大学 情報科学研究科)

E.1.1 ファイルの準備

後述のフォーマットに従った正解単語列ファイルと仮説単語列ファイルを準備する。Julius の認識結果ログから仮説単語列ファイルを作成する場合には、mkhyp.pl が使用できる。

E.1.2 アラインメント

正解単語列ファイルと仮説単語列ファイルを読み込んで、DP を用いて両者のアラインメント (align.pl) を取る。ここでは、単位として形態素単位か文字単位、表記として漢字かな混じり表記かカタカナ表記のどちらかを指定してアラインメントを取ることができる。また、形態素単位でアラインメントを取る際には、1 対多の複合語処理を行うかどうかを指定できる。ただし、正解単語列ファイルと仮説単語列ファイルは後述のフォーマットに従っている必要がある。アラインメントを取った後、各単語ペアに対して正解、置換誤り、挿入誤り、脱落誤りいずれかのラベルが付与される。

E.1.3 スコアリング

アラインメント結果に基づいてスコアリング (score.pl) を行う。スコアリング結果は、(1) 各話者の発話文ごとのスコアリング結果、(2) 各話者ごとのスコアリング結果詳細、(3) システム全体のスコアリング結果、(4) システム全体のスコアリング結果詳細の 4 段階に分けて報告される。これは、NIST BENCHMARK SPEECH RECOGNITION SYSTEM SCORING PROGRAM[1]を参考にしている。

E.2 正解・仮説単語列のファイルフォーマット

正解単語列ファイルと仮説単語列ファイルは、id 番号と単語列で構成されていて、id 番号の行の次に、対応する単語列の行がくる形になっている必要がある。id 番号と単語列は以下のフォーマットに従わなければならない。

- id 番号 ... 半角アルファベットまたは数字からなる話者番号と文番号がハイフンで区切られた形になっていること。

  例）`nm005-035`

[1] ftp://jaguar.ncsl.nist.gov/pub/score3.6.2.tar.Z

- 単語列 ... 各単語が半角スペースで区切られた形になっていること。なお、各単語は、形態素の表記、読み、原形 (表記と原形が同じ場合は省略)、品詞番号類 (品詞番号、活用型、活用形が'/' 記号で結合されたもの、活用型、活用形がない場合は省略してもよい) が'+' 記号で結合された形になっている必要がある。

  例)␣個人技+コジンギ+2␣が+ガ+58␣随所+ズイショ+2␣で+デ+58␣光っ+ヒカッ+光る+44/17/8␣た+タ+70/47/2␣。+。+74

また、正解単語列ファイルと仮説単語列ファイルは、話者番号と文番号でソートされていなければならない。

以上のフォーマットに沿った正解 (仮説) 単語列ファイルの一部を例として示す。

正解 (仮説) 単語列ファイルの例

```
nm005-025
お+オ+39␣年寄り+トシヨリ+2␣から+カラ+58␣の+ノ+67␣注文+チューモン+17␣に+ニ
+58␣も+モ+62␣備え+ソナエ+備える+44/6/5␣、+、+75␣二千+ニセン+19␣個+コ+33␣分+
ブン+28␣の+ノ+67␣材料+ザイリョー+2␣を+ヲ+58␣確保+カクホ+17␣。+。+74
nm005-035
個 人 技+コ ジ ン ギ+2␣が+ガ+58␣随 所+ズ イ ショ+2␣で+デ+58␣光っ+ヒ カッ+光
る+44/17/8␣た+タ+70/47
/2␣。+。+74
                    .
                    .
                    .

nm006-020
ラジオ+ラジオ+2␣番組+バングミ+2␣の+ノ+67␣ゲスト+ゲスト+2␣として+トシテ+60␣
迎え+ムカエ+迎える+44/6/5␣た+タ+70/47/2␣時+トキ+22␣の+ノ+67␣こと+コト+21␣
です+デス+70/49/2␣。+。+74
                    .
                    .
                    .
```

E.3 各プログラムの説明および使用法

ここでは、本ツールに含まれている各プログラムの簡単な説明と使用法について述べる。全てのプログラムは Perl あるいは JPerl で書かれているため、プログラムを使用するには Perl と JPerl がインストールされていることが必要となる。また、Perl と JPerl がインストールされているディレクトリにコマンドサーチパスが通っていなければならない。

E.3.1 mkhyp.pl

mkhyp.pl は、Julius(ver 1.2) の認識結果ログから、フォーマットに従った仮説単語列ファイルを作成するためのプログラムである。

Julius(ver 1.2) の認識結果ログを標準入力から読み込み、仮説単語列を標準出力に返す。プログラムは EUC コードで書かれているので、ログファイルを EUC コードに変換する必要がある。

-p オプションで Julius のどちらのパスの仮説単語列ファイルを作成するか指定する。'-p 1' とすると第 1 パス、'-p 2' とすると第 2 パスの仮説単語列ファイルを作成する。

使用法

```
% nkf -e 認識結果ログファイル名 | \
  jperl -Leuc mkhyp.pl \
      -p (1|2) \
  > 仮説単語列ファイル名
```

E.3.2 align.pl

align.pl は、正解単語列ファイルと仮説単語列ファイルのアラインメントを取って、各単語ペアに対して正解、置換誤り、挿入誤り、脱落誤りの判定を行うためのプログラムである。

仮説単語列ファイルを標準入力から読み込み、アラインメント結果を標準出力に返す。正解単語列ファイルは-r オプションの後に指定する。

-u オプションでアラインメントの単位を指定する。'-u morpheme' で形態素単位、'-u char' で文字単位でアラインメントを行う。形態素単位でアラインメントを取る際、-c オプションを同時指定することにより 1 対多の複合語処理を行うことができる。

-f オプションでどの表記でアラインメントを取るか指定する。'-f kanji' で漢字かな混じり表記、'-f kana' でカタカナ表記でアラインメントを行う。

使用法

```
% jperl -Leuc align.pl \
      -u (morpheme|char) \
      [-c] \
      -f (kanji|kana) \
      -r 正解単語列ファイル名 \
      仮説単語列ファイル名 \
  > アラインメントファイル名
```

E.3.3 score.pl

score.pl は、アラインメント結果に対してスコアリングを行うプログラムである。

アラインメントファイルは標準入力から読み込まれる。スコアリング結果は、(1) 各話者の発話文ごとのスコアリング結果、(2) 各話者ごとのスコアリング結果詳細、(3) システム全体のスコアリング結果、(4) システム全体のスコアリング結果詳細の 4 段階に分けて作成される。(1) は” 話者 id.snt”、(2) は” 話者 id.sum”、(3) は” アラインメントファイル名.sys”、(4) は” アラインメントファイル名.sys_dtl” というファイルに各々出力される。なお、これらの結果は” アラインメントファイル名.scr” というディレクトリに作成される。

使用法

```
% perl score.pl アラインメントファイル名
```

付 録F　音声認識辞書作成ツール説明書†

vocab2htkdic.pl は、CMU SLM Toolkit の語彙ファイル (.vocab) から音声認識用辞書を作成するプログラムである。

作成する音声認識用辞書は、以下の例に示すように、HTK の dictionary format とほぼ同等のフォーマットになっており、第 1 フィールドが単語名、第 2 フィールドが出力シンボル、第 3 フィールドが音素 HMM の並びになっている。第 3 フィールドの作成には、単語名の読みの部分を用いて音素列を展開するようになっているので、各単語は、形態素の表記、読み、原形 (表記と原形が同じ時は省略)、品詞番号類が'+' 記号で結合された形になっていなければならない。

音声認識用辞書の例

```
        .
        .

あいさつ+アイサツ+17  [あいさつ]  a i s a ts u
あいつ+アイツ+14      [あいつ]    a i ts u
あいまい+アイマイ+18  [あいまい]  a i m a i
あう+アウ+44/21/2     [あう]      a u

        .
        .
```

次にこのプログラムの使用法について説明する。プログラムは JPerl で書かれており、文字コードは EUC コードになっている。

† 山本 俊一郎 (奈良先端科学技術大学院大学 情報科学研究科)

使用法

```
% nkf -e 語彙ファイル | \
  jperl -Leuc vocab2htkdic.pl \
      [-o 出力ファイル名] \
      [-e エラーリストファイル名] \
      [-u (ngram|keitaiso|yomi)] \
      -p 音素変換リストファイル名
```

-o オプションでは、認識辞書を出力するファイル名を指定する。オプション指定を行わなかった場合は、”HTKDIC” というファイルに出力される。

-e オプションでは、認識辞書を作成する際に音素列に正しく変換できなかったエントリのリストを出力するファイル名を指定する。オプション指定を行わなかった場合は、”ERROR” というファイルに出力される。変換に失敗したエントリについては、このエラーリストを見て修正する。エラーリストの例を以下に示す。行頭の数字は、認識辞書の行数を表している。

エラーリストの例

```
38: 「 ’ +未定義語+17 [「 ’ +未定義語+17] 未定義語
1348: っ+ッ+く+45/9/7 [っ+ッ+く+45/9/7] q
2166: ょ+未定義語+17 [ょ+ョ+17] ョ
```

-u オプションでは、認識辞書の第 2 フィールドの [ ] 内のフォーマットを指定する。’-u ngram’ とすると、[ ] 内には N-gram の単位 (形+読+原+品) が、’-u keitaiso’ とすると、[ ] 内には漢字かな混じりの表記が、’-u yomi’ とすると、[ ] 内には読みの表記のみが入る。指定しなかった場合は、[ ] 内には N-gram の単位 (形+読+原+品) が入る。なお、’-u ngram’ とした場合、併記読みについては、以下の例のように、対応する音素列ごとに [ ] 内に入る読みを展開する。

併記読みの展開例

```
—+{—/カラ/タイ/ヒク}+73/0/0   [—+—+73/0/0]  sp
—+{—/カラ/タイ/ヒク}+73/0/0   [—+カラ+73/0/0]       k a r a
—+{—/カラ/タイ/ヒク}+73/0/0   [—+タイ+73/0/0]       t a i
—+{—/カラ/タイ/ヒク}+73/0/0   [—+ヒク+73/0/0]       h i k u
```

-p オプションでは、音素変換規則のリストファイルを指定する (必須)。音素変換規則のリストファイルは、以下の例に示すように、カナと音素列が’ +’ 記号で区切られたフォーマットになっている。また各音素の末尾には半角スペースが必要なので注意すること。

kana2phone_rule.ipa が IPA 用の音素変換規則リストファイルなので、これを指定する。

音素変換リストファイルの例

```
      ・
      ・

リャ+ry␣a␣
リュ+ry␣u␣
リョ+ry␣o␣
ギャ+gy␣a␣
ギュ+gy␣u␣
ギョ+gy␣o␣

      ・
      ・
```

付 録G　数字表現正規化ツール説明書†

suuzi_syori.pl は、ChaWan を通した後の ChaSen 解析結果に対して、数字表現の正規化を行なうプログラムである。ChaSen 解析結果は、以下に示す例のように、形態素の表記、読み、原形、品詞番号類 (品詞番号、活用形、活用型が'/' 記号で結合されたもの) がタブスペースで区切られた形式になっている必要がある。

ChaWan を通した後の ChaSen 解析結果の例

```
１２月　ジューニガツ　　１２月　16/0/0
は　　ワ　　は　　62/0/0
３８・６　　サンジュー{ハチ/ハッ}テン{ロク/ロッ}　　３８・６　　19/0/0
％　　パーセント　　％　　33/0/0
と　　ト　　と　　64/0/0
少数　　ショースー　　少数　　2/0/0
EOS
```

数字表現の正規化では、以下の例のように、品詞番号類が'19/0/0' である行に対して、(1) 表記を漢数字に統一、(2) 位取り単位に連続する数字を展開する。また、品詞番号類が'16/0/0' である行で、数字と副詞可能の名詞が固められている語のうち、月の表現については、数字を漢数字に統一する。ただし、位取りは行わない。

† 山本 俊一郎 (奈良先端科学技術大学院大学 情報科学研究科)

数字表現の正規化の例

```
十二月  ジューニガツ    十二月  16/0/0
は      ワ      は      62/0/0
三十    サンジュー      三十    19/0/0
八      {ハチ/ハッ}     八      19/0/0
・      テン    ・      19/0/0
六      {ロク/ロッ}     六      19/0/0
%       パーセント      %       33/0/0
と      ト      と      64/0/0
少数    ショースー      少数    2/0/0
```

次にこのプログラムの使用法について説明する。プログラムは JPerl で書かれており、文字コードは EUC コードになっている。

使用法

```
% nkf -e ChaSen 解析結果ファイル名 | \
  jperl -Leuc suuzi_syori.pl
      -l 数詞の正規化リストファイル名 \
      -r 副詞可能名詞の正規化リストファイル名 \
  > 出力ファイル名
```

品詞番号類が'19/0/0' の数字表現については、オプション-l で指定する数詞の正規化リストを参照して正規化を行なう。品詞番号類が'19/0/0' の月表現については、オプション-r で指定する副詞可能名詞の正規化リストを参照して正規化を行なう。新たに処理対象にしたい数表現が出現した場合には、これらのリストに追加すればよい。2 つのリストの一部を以下に示す。

数詞の正規化リストファイルの一部 (suuzi_syori.list より)

・
・

イチ␣一
イッ␣一
ツイタ␣一
ヒト␣一
ワン␣一
{イチ/イッ}␣一
{イチ/イッ/ヒト}␣一
{イッ/ヒト}␣一

・
・

副詞可能名詞の正規化リストファイル (suuzi_syori.ref より)

イチガツ␣一月
ニガツ␣二月
サンガツ␣三月
シガツ␣四月
ゴガツ␣五月
ロクガツ␣六月
シチガツ␣七月
ナナガツ␣七月
ハチガツ␣八月
クガツ␣九月
ジューガツ␣十月
ジューイチガツ␣十一月
ジューニガツ␣十二月