演習 3 辻井研
第二回 単語クラスタリングの諸考察

岡野原　大輔

2004/05/31

1　N-gram Model

単語の出現予測に直前の (N-1) 単語の情報のみを用いる言語モデルを N-gram Model という（以降 [2]）。単語が N 単語より前にある単語の影響を全く受けないこの大胆な仮説は、多くの実験でその有用性が実証されている。N-gram Model は単語の生起を N-1th order Markov Model で近似したモデルである。

N の値が大きくなればなるほど、推定するパラメータ数が指数的に増大するため通常は $N = 2, 3$ を用いる。（$N = 4$ 以降 N を大きくしても精度は向上しないことが知られている。）$N = 0, 1, 2, 3$ の N-gram をそれぞれ、zerogram、unigram、bigram、trigram という。

学習データが増えれば増えるほど、N-gram Model は精度が向上するが、学習データ中に出現しない単語組が非常に多くあることも知られている。このため、確率値を直接、相対頻度により推定する方法では、学習データ中に出現していない単語組の確率を 0 にしてしまったり、出現回数の少ない確率の推定精度が極端に低くなるなどの問題（sparseness problem）がある。このため、通常は smoothing を用いて単語の推定確率を補正し、学習データ中に出現していない、または出現回数が少ない単語組に適切な確率を与える必要がある。

N-gram Model は次のように定式化され、最尤推定である相対頻度から、確率は次のように推定することができる。なお、最尤推定でもとめられた確率を P_{ML} とする。

$$P(w_n|w_1^{n-1}) = P(w_n|w_{n-N+1}^{n-1}) = P_{ML}(w_n|w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n})}{C(w_{n-N+1}^{n-1})} \tag{1}$$

与えられた単語列 w_1^n が N-gram Model によって生成される確率は乗法定理を用いて次のように定式化できる。(文頭、文末には特殊な単語が出現していると考える）

$$P(w_1^n) = \Pi_i P(w_i|w_{i-N+1}^{i-1}) \tag{2}$$

smoothing は、様々な方法があるが代表的なものを取り上げる [1]。

- additive smoothing

$$P_{add}(w_i|w_{i-N+1}^{i-1}) = \frac{\sigma + C(w_{i-N+1}^{i})}{\sigma|V| + \Sigma_{w_i} C(w_{i-N+1}^{i})} \tag{3}$$

- Jelinek-Mercer Smoothing

$$P_{interp}(w_i|w_{i-N+1}^{i-1}) = \lambda_{w_{i-N+1}^{i-1}} P_{ML}(w_i|w_{i-N+1}^{i-1}) + (1 - \lambda_{w_{i-N+1}^{i-1}}) P_{interp}(w_i|w_{i-N+2}^{i-1}) \tag{4}$$

今回は、Fixed Kneser-Ney Smoothing を用いる。これは次のように定式化される。

- Fixed Kneser-Ney Smoothing

  最高次オーダー

$$P_{KNfix}(w_i|w_{i-N+1}^{i-1}) = \frac{max(C(w_{i-N+1}^{i}) - D, 0)}{\Sigma_{w_i} C(w_{i-N+1}^{i})} + \frac{D}{\Sigma_{w_i} C(w_{i-N+1}^{i})} N_{1+}(w_{i-N+1}^{i-1}*) P_{KN_fix}(w_i|w_{i-N+2}^{i-1}) \quad (5)$$

  それ以外のオーダー

$$P_{KNfix}(w_i|w_{i-N+1}^{i-1}) = \frac{max(N_{1+}(w_{i-N+1}^{i}) - D, 0)}{\Sigma_{w_i} N_{1+}(w_{i-N+1}^{i})} + \frac{D}{\Sigma_{w_i} N_{1+}(w_{i-N+1}^{i})} N_{1+}(w_{i-N+1}^{i-1}*) P_{KN_fix}(w_i|w_{i-N+2}^{i-1}) \quad (6)$$

  ただし、$n1$、$n2$ をそれぞれ、一回、二回だけ出現した N-gram の個数としたとき、

  $D = \frac{n_1}{n_1+2n_2}$

  $N_{1+}(w_{i-N+1}^{i-1}*) = |w_i : C(w_{i-N+1}^{i-1} w_i) > 0|$

2 Class Model

前回説明したように、単語を分類することは、言語モデルの精度向上、計算量削減につながる。今回作成する Class Model[3] は、Class Model による学習データの対数尤度の最適化という観点から単語を Class に分類する。最初に次のようなバイグラム・クラスモデルを説明する。これは単語の生成確率が、その単語のクラスと直前の単語のクラスによる bi-gram、及びクラスから単語の生成確率の積になっている。

$$P_{cm}(w_n|w_{n-1}) = P(w_n|c_n)P(c_n|c_{n-1}) \quad (7)$$

上記のバイグラム・クラスモデルによる、学習データ w_1^T の対数尤度 $L(\pi)$ は以下のように計算できる。

$$L(\pi) = \Sigma_{i=1}^{T} log P_{cm}(w_n|c_n)P(c_n|c_{n-1}) \quad (8)$$

$$= \Sigma_{i=1}^{T} log \frac{C(w_i)C(c_{i-1}c_i)}{C(c_i)C(c_{i-1})} \quad (9)$$

$$= \Sigma_{c_1,c_2} C(c_1,c_2) log \frac{C(c_1,c_2)}{C(c_1)C(c_2)} + \Sigma_w C(w) log C(w) \quad (10)$$

右辺第一項のみがクラスに依存するので、この変化を調べることにより、最適な単語のクラス分類を求めることができる。クラス分類については次回説明する。

クラスモデルを N-gram の補間に用いて言語モデルの精度向上をはかることができる。例えば、Tri-gram(P_{bi}) と Bi-gram Class Model(P_{bcm}) を補間した場合、次の式となる。

$$P(w_n|w_{n-N+1}^{n-1}) = \lambda P_{bi}(w_n|w_{n-2}w_{w-1}) + (1-\lambda) P_{bcm}(w_n|w_{n_1}) \quad (11)$$

3 実装状況と今後

今週は、テスト環境の実装をおこなった。Corpus は British National Corpus（以下 BNC) を用いる。

BNC から、必要な情報のみを取り入れるモジュールを作成した。これは、コーパスから、タグ情報、及び句読点などを取り除き、単語列のみに変換するモジュールである。変換されたコーパスから、Tri-gram Model を構築する。この際、Smoothing には上記に挙げた Fixed Kneser-Ney Smoothing を用いる（現状では、

Absolute Discounting を実装)。こうして構築した言語モデルを用いて、テストデータの Perplexity を求める部分まで構築した。また、Class Model を構築する部分はそのまま用いることができた。

今後は、このテスト環境を元に、各種アイディアを実装して、テストを行う予定である。

参考文献

[1] Chen, S. F. Goodman, J. 1996, 1998. An empircal study of smoothing techniques for language modeling. In Proceedings of the 34th Annual Meething of the Association for Computational Linguistics (ACL-96) pp.310-318.

[2] 北 研二 1999　言語と計算 4　確率的言語モデル　東京大学出版会 ISBN:4-13-065404-7

[3] Martin, S., Liermann, J. and Ney, H. 1998. Algorithms for bigram and trigram word clustering. Speech Communication 24(1998) 19-37