1L-05

単語 ID の remapping によるダブル配列言語モデルの効率化

幡鉾 勇哉† 山本 幹雄‡

筑波大学 情報学群 情報メディア創成学類† 筑波大学 システム情報系‡

1 はじめに

ダブル配列言語モデル[1][2]は,ダブル配列[3]を用いた Ngram 言語モデルの効率的な実装である.大きなテキストデータから学習した Ngram 言語モデルは高性能であるが,記憶する Ngram の種類数が多くなり,ダブル配列のサイズ・構築速度を悪化させる.本稿では単語 ID を Ngram のレベルによって変化させる'Remapping'と呼ばれる手法[4]をダブル配列に適用することを提案し,サイズ・構築速度の効率を改善できることを示す.

2 ダブル配列言語モデル

Ngram 言語モデルは,Ngram と呼ばれる N 個の単語連鎖を入力としてその条件付き確率を返すモジュールである.このため,表探索問題に対する解が基本的な言語モデルの実装手法となる.表探索問題には様々な解法があるが,言語モデルの場合はトライ(TRIE)とハッシュ法による実装が主流である.ダブル配列言語モデル(以下,DALM(Double-Array Language Model)と略す)は,TRIE のコンパクトかつ高速な実装であるダブル配列[3]をベースとし,言語モデルの性質を最大限利用することでバランスのよい実装を実現している[1][2].

図 1 に示すように,ダブル配列は TRIE の(疎)行列表現を 2 本の配列で表現したデータ構造であり,(疎)行列表現の高速性を保ったままコンパクトにできる[3].(疎)行列表現は TRIE のノードを行に割り当て,列に遷移記号の種類(数字で表される)を割り当て,ノードから子ノードへの遷移先(子ノードの行番号)を行で表現したものである.しかし,Ngram 言語モデルに応用した場合,子ノードの数が冪分布するため,行列は疎となり,全体的なサイズは非現実的となる.この無駄をなくすために,列方向にぶつからないようにすべての行をずらして 1 本にまとめ(next 配列),さらに誤遷移をなくすために親ノードの情報を別の 1 本の配列に格納する(check 配列).行をずらした長さを記録した offset 配列を加えて 3 本の配列で表現する方法がトリプル配列である.さらに,offset 配列の値を next 配列に代入し(base 配列),check 配列を親ノードの index 値に置き換えたものがダブル配列であり,base 配列と check 配列の 2 本で(疎)行列を表現する.

TRIE が巨大になった場合,列方向にぶつからないようするために大きなずらしが必要となる.結果的に隙間が多くなり配列(モデルサイズ)が大きくなってしまうという問題がある.これは単語 ID を出現数の大きい単語から順に付与することによって多少は改善されるが,次節でさらに改良する方法を提案する.

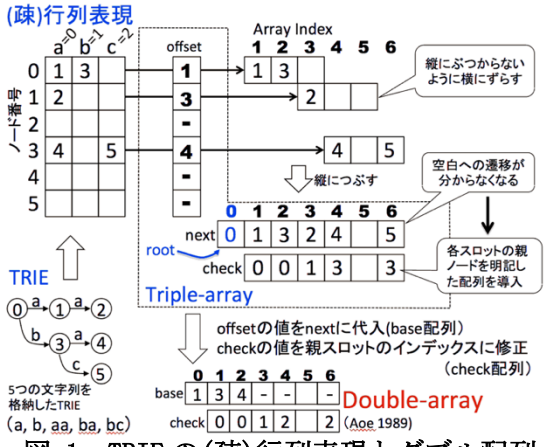

図 1 TRIE の(疎)行列表現とダブル配列

3 Remapping とダブル配列への適用

Remapping は各ノードから子ノードへ分岐する単語 ID 番号を付け替えて,子ノード集合への分岐の ID の幅(分岐する可能性のある単語 ID 番号の**範囲**)を小さくする手法である[4].

以下,Pibiri らの論文の図に準じて作成した図 2 によって,Remapping による単語 ID の付け替えを説明する.図 2 は単語が 4 種(a, b, c, d)で,最大 3 階層の TRIE である.最上層の丸の中の数字がデフォルトの単語 ID である.3 階層目の左下の丸の中の数字が Remapping によって付け替えられた単語 ID である.Remapping は単語 ID をその直前の単語を条件とした Remapping 用の辞書を参照して単語 ID を付け替える.Remapping 用の辞書は,TRIE の 1 階層目の単語を条件とし,2 階層目に現れる単語に 0 から順に ID を付与することによって作る(2 階層の右上の数字).例えば,1 階層目の単語'b'に続く単語は'a','c','d'の 3 つであり,それぞれ 0, 1, 2 の数字を割り当てる.この単語'b'に条件

Improving efficiency of double-array language models by remapping word IDs
Yuya Hatahoko and Mikio Yamamoto
University of Tsukuba

付けられたRemapping用辞書を用いて,単語'c'→単語'b'の下の3階層目の単語'a'と'd'をそれぞれ0と2に付け替える. このようにTRIE木の子ノード集合への分岐ID幅は必ず小さくなる.これによって,比較的容易にダブル配列を構築できるようになるというのが本稿のアイデアである.

図2は直前の単語1つを条件とした場合のRemapping(これを'unigram文脈'でのRemappingと呼ぶ)であるが,直前の2単語 (これを'bigram文脈'でのRemappingと呼ぶ)あるいは3単語のように条件を細かくすれば,単語IDの幅をより狭くできる.

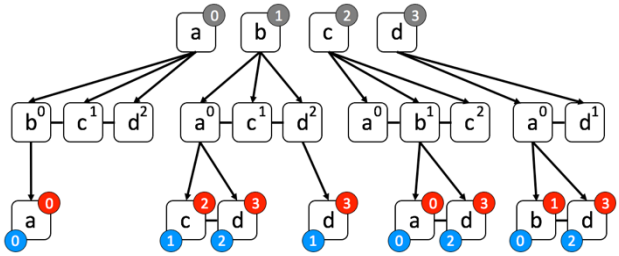

図2 Remmappingによる単語IDの付け替え

4 評価実験

4.1 評価方法と評価データ

Remapping しない場合とした場合について,サイズが4種類の5gram言語モデルデータ (Ngramの種類数がそれぞれ100M,200M,500M,1G) を用いて,ダブル配列の長さと構築時間を比較した.Remappingする場合は次の3種類の手法を試みた.

- unigram文脈:TRIEの1-2階層目を辞書として,3-5階層目をremapping
- bigram文脈:TRIEの1-3階層目を辞書として,4-5階層目をremapping
- uni+bi文脈:TRIEの1-2階層目を辞書として,3階層目をremapping,かつTRIEの1-3階層目を辞書として4-5階層目をremapping

ダブル配列の長さはbase配列とcheck配列で長さが異なる.5gramモデルの場合,TRIEの5階層目は子ノードを持たないため,5階層目のノードはbase配列のセルが必要ない.このため,base配列の後半に5階層目のノードが集まるように工夫することによってbase配列の後半部分を削除できる[2].一方,check配列はすべてのノードに対して必要である.実験ではこの2つの配列の長さをそれぞれ比較した.

4.2 評価結果

図3はRemappingしない場合とRemappingした場合のモデルサイズおよびモデル構築時間を比較した結果である.上がcheck配列に対する結果, 下がbase配列に対する結果である.縦軸が配列の長さであり,横軸が構築時間である.配列の長さはNgramの総数(隙間なく並べた場合の長さ)に対する比率で表している.

Remappingしない場合に比べて,unigram文脈,bigram文脈,両方の文脈の順に圧縮率が高まっていることが確認できる.また,Ngramの種類数が多くなるほど圧縮効果が高まっており,1Gのデータに対して両方の文脈を用いた場合30%ほどの圧縮を達成しており,ほぼ限界まで圧縮できていることがわかる.加えて,構築速度もわずかであるが早くなっていることが確認できる.

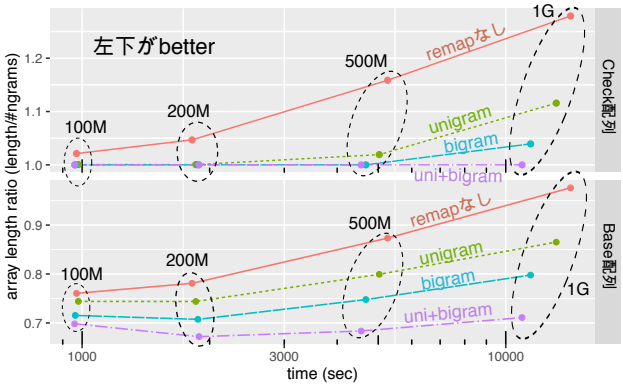

図3 remmappingの効果

5 おわりに

Remappingによる単語IDの付け替えによってダブル配列言語モデルの圧縮率を高められることを確認した. 特にNgram言語モデルのサイズが大きくなるほど効果を発揮しているのでスケール効果が期待できる. 今後の課題はRemapping用の辞書のサイズや検索速度の低下に関する検討である.

参考文献

[1] J.Norimatsu et al., "A fast and compact language model implementation using double-array structures," ACM Transaction on Asian and Low-Resource Language Information Processing, 15(4), 2016.

[2] 竹中,芳賀,山本,「部分転置ダブル配列を用いたngram言語モデルの実装」, 言語処理学会第23回年次大会発表論文集, pp.663-666, 2017.

[3] J. Aoe, "An efficient digital search algorithm by suing a double-array structure," IEEE Transactions on Software Engineering, 15(9), pp.1066-1077, 1989.

[4] G.E.Pibiri and R.Venturini, "Efficient data structures for massive n-gram datasets," in the proc. of SIGIR2017, pp.615-624, 2017.