4U-06

双方向配置によるコンパクトかつ高速なダブル配列言語モデル構築

石井瑛彦 † 山本 幹雄 ‡

†筑波大学大学院システム情報工学研究科 ‡筑波大学システム情報系

1 はじめに

トライ (Fredkin et al., 1960) で表現された ngram 言語モデルの効率の良い実装法としてダブル配列 (Aoe, 1989) を用いた DALM(Double Array Language Model) (Norimatsu et al., 2016) が提案されている．しかし DALM が他の実装法に比べ構築に膨大な時間がかかる (Nikolay et al., 2016). これはダブル配列構築においてトライのノードの子ノード群を配列の空き要素に配置する処理がありその配置場所探索処理の最悪計算量がトライのノード数に対して 2 乗のオーダーとなっているためである．高速化の工夫として入力トライの分割により計算量を削減し高速化する手法があるが分割数が増えると子ノード数の多いノードを配置してできた隙間を埋めるような子ノード数の少ないノードが分散してしまい空き要素の多いダブル配列ができやすくなりモデルサイズが大きくなってしまう．本研究は分割による高速化を前提とし, コンパクトさを維持したまま構築時間を短縮することを目的とする．

本稿では, 配置場所を順方向だけでなく逆方向にも探索する双方向配置を提案し構築実験によりコンパクトさを維持したまま高速化できることを示す．一方で双方向配置された言語モデルはクエリ処理時に遷移先ノードがどちらの方向に配置されているか分からず両方向を探索する必要があり検索スピードが落ちてしまうことが懸念されるが本研究では翻訳実験とパープレキシティ (以下 ppl) 計算実験により検索スピードの低下が翻訳時には僅かで済むことを示した．

2 DALM 構築

ダブル配列はスパースになりがちなトライの遷移表を空き要素の少ない 2 本の配列で表現でき, 配列構造による高速な動作とコンパクトさを両立した効率の良い実装法である. 図 1 はトライからダブル配列を構築する処理の流れを表している．まずトライの遷移表の各行を情報を持つ要素が縦に重ならないようにずらし next という配列に配置する。この時, 遷移のための情報としてそれぞれの行のずらし幅を offset 配列に格納し遷移が成功していることを確認するために親ノード (遷移元) の情報を check 配列に格納する。さらに next 配列の値をノードに対応する offset 値で書き換え新たに base 配列とする．さらに check 配列の値を next 配列の対応するノード ID の配列 index で書き換える．こうしてトライのノードの遷移情報を疎な遷移表から密な 2 本の配列で表現できる．ダブル配列構築処理において最も時間のかかる処理はノードの子ノード列 (遷移表の行の値を持つ要素) を他のノードの子ノード列とぶつからないように 1 本の配列に配置する処理で

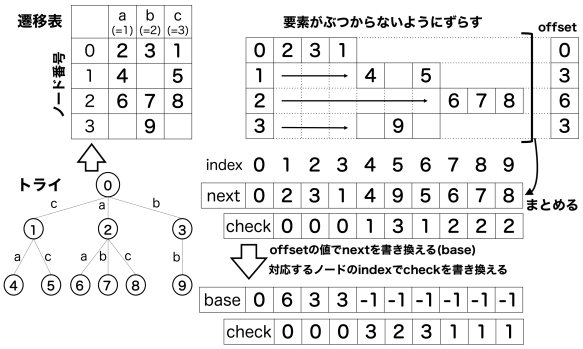

図 1: DALM 構築

あり, 配置場所探索の最悪計算量はノード数に対して 2 乗のオーダーとなる．これを高速化するための手法として代表的なものがトライの分割である．分割数を D として分割した場合各トライのノード数は $1/D$ になるのでその計算量は $1/D^2$ となり大幅な構築時間の短縮が見込める．しかし分割数を増やすと空き要素が増えやすくモデルサイズが肥大化してしまう。これは子ノード数の多いノードが配置された際にできる隙間を子ノード数の少ないノードが埋めていたのが分割によりノードが分散し埋めきれなくなったからであると考えられる．そこで本研究では子ノード数が多いノードでも隙間を埋められるようにする手法として双方向配置を提案する。

3 双方向配置

双方向配置はこれまでノードの子ノードの配置場所探索を順方向 (index が大きくなる方) のみとしていた所を逆方向も探索し可能な限り逆方向に配置することで子ノード数の多いノードでもすでに配置された配列の空き要素を埋めることができるようにすることで分割数の増加に伴う配列長の肥大化を防ぐという手法である．図 2 は従来の手法と提案手法の違いを表している．図 2 はすでにノードが配置されたダブル配列に遷移表のある行を配置をする流れを示している．配置場所の探索は各ノードの 1 つ目の子ノードをダブル配列上の空き要素がある場所までずらしその他の子ノードが全て配置可能であるかを調べ, 可能であれば配置しすでに配置されたノードとぶつかってしまうのであれば 1 つ目の子ノードを次の空き要素までずらすという作業を配置できるまで繰り返す．提案手法により従来では配置できなかったずらし幅でノードが配置可能となり, 配列長を伸ばすことなく配置できる上に, 隙間を埋められることが分かる．さらに提案手法には構築時間を短縮できる可能性がある．理由としては, あるずらし幅で配置可能かどうかを確かめる処理の計算量は倍になるが配置可能になる可能性が高くなるため空き要素を辿る回数が減りが減り全体として計算量の削減に繋がる場合があるためで

Fast and compact double array language model construction with bidirectional search

†Akihiko ISHII ‡Mikio YAMAMOTO

†Graduate School of Systems and Information Engineering, University of Tsukuba

‡Faculty of Engineering, Information and Systems, University of Tsukuba

Copyright ©2019 Information Processing Society of Japan.
All Rights Reserved.

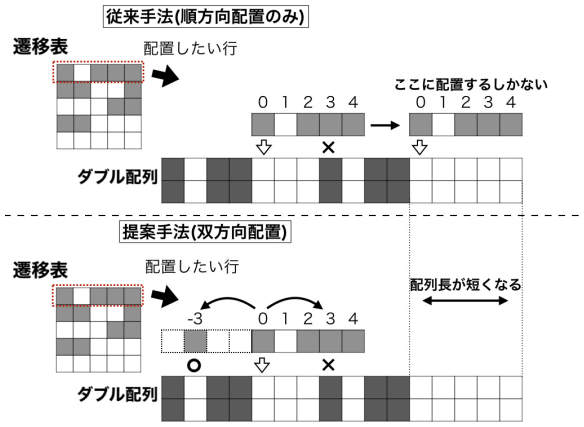

図 2: 双方向配置

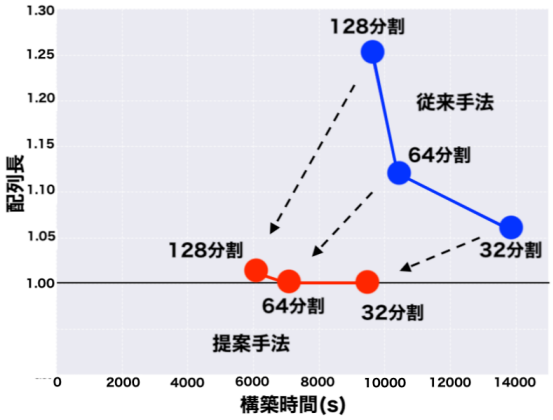

図 3: 構築実験による比較

ある．一方で双方向配置された DALM を用いて検索をする場合これまで順方向にのみ遷移して探索すればよかったものが逆方向への遷移も確かめなくてはいけなくなり検索スピードが低下すると考えられる．以上を踏まえ本手法を構築時間, モデルサイズ, クエリ処理速度の 3 点で評価しその有効性を実験により検証する．

4 実験

提案手法の有効性を構築実験と翻訳・ppl 計算時間の測定実験により示す．

4.1 構築実験

構築実験により構築時間と, モデルサイズを評価する．入力となるトライは, ネット上のニュースサイト記事の文章を学習したエントリー数 (ノード数) が 10 億のものを用いた．実験に用いたマシンの性能は CPU が Xeon E5-2620v4 2.10GHz, メモリ 256GB, コア数 16 となっている．従来手法, 提案手法それぞれで分割数を 32,64,128 と変化させ構築実験を行いその構築時間とモデルサイズを比較する．なお DALM 構築は並列化に対応しており本実験では並列数を 16 とした．

4.2 検索スピードの測定実験

双方向配置によるモデルの検索スピードの低下を検証するため翻訳実験とテストセットによる ppl 計算を行いそれぞれにかかった時間を計測した．翻訳対象には特許文の英日対訳コーパスの英文 1381 文を用い, テストセットには学習に用いなかったニュース記事コーパスから作成した 1 万文のテキスト (以下 Testset_10k とする) と, 残った 16,679,044 文のテキスト (以下 Testset_others) を用いた．全てのテキストは互いに独立しており学習データにも含まれていない．翻訳実験は言語モデルのクエリ処理以外にも様々な処理があるた言語モデルの検索速度による差は出にくいが実用面での性能を評価できる．対して ppl 計算は言語モデルによるクエリ処理のみが行われるので純粋な検索スピードの比較ができる．

4.3 実験結果と考察

図 3 は構築実験の結果を表しており横軸を構築時間 (秒), 縦軸をモデルサイズを表す値として配列長と配置したノード数 (10 億) の比率としている．配列長の比率は配列長をノード数で割った値としており隙間なく配置できた場合は 1.0 となり隙間が多いと値が大きくなっていくのでグラフでは左下に行くほど構築時間が短くサイズの小さな良いモデルであると言える．図 3 より従来手法では分割数を増やすと構築時間は短くなるがモデルサイズが大きくなることが分かる．それに対して提案手法は分割数が増えてもモデルサイズが最小 (1.00) からほとんど伸びずさらに従来手法よりも構築時間を短縮できることが分かった．翻訳実験の結果は従来手法が 4112.90 秒に対し提案手法は 4101.76 秒となった．ppl 計算実験は Testset_10k では従来手法が 143 ミリ秒, 提案手法が 161 ミリ秒となり Testset_others では従来手法で 244460 ミリ秒, 提案手法は 271637 ミリ秒となった．ppl 計算実験より検索スピードは 1 割ほど遅くなることが分かった．構築実験の際に逆方向へ配置されたノードを数えた結果 10 億ノードのうち約 1.8 億ノードが逆方向に配置されていたためこの結果は妥当なものと言える．しかし翻訳実験の結果からその差は翻訳スピードにはあまり影響しないことが分かった．以上実験より双方向配置により構築の高速化とモデルサイズの短縮を両立することができ, 懸念された検索スピードの低下も実用面での影響がほとんどないことが分かり提案手法の有効性を示した．

5 おわりに

本稿では DALM 構築における分割数の増加に伴うモデルサイズの肥大化を抑えつつ構築時間を短縮できる手法として配置場所の探索を逆方向にも行う双方向配置を提案し構築実験によりその効果を示した．さらにアルゴリズム上懸念された検索スピードの低下を翻訳実験と ppl 計算実験により検証し翻訳のスピードにはあまり影響しないことを明らかにした．今後はより逆方向に配置されやすいような手法を検討して行く．

参考文献

Aoe, Jun-ichi (1989).“An efficient digital search algorithm by using a double-array structure,” *IEEE Transactions on Software Engineering*, Vol. 15, No. 9, pp. 1066–1077.

Norimatsu, J., M. Yasuhara, T. Tanaka, and M. Yamamoto (2016).“A fast and compact language model implementation using double-array structures,” *ACM TALLIP* Vol. 15, No. 4, 27 pages.

Fredkin, Edward (1960). “Trie Memory”,Communications of the ACM, Vol. 3, No. 9, pp. 490–499.

Nikolay Bogoychev, Adam Lopez (2016). “N-gram language models for massively parallel devices.”,*In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pp. 1944–1953, August 7-12.

Copyright ©2019 Information Processing Society of Japan.
All Rights Reserved.