逐次語彙選択によるニューラル Input Method の高速化

姚 佳立
jiayao@microsoft.com
マイクロソフト

朱 中元
shu@nlab.ci.i.u-tokyo.ac.jp
東京大学 大学院情報理工学系研究科

李 心剣
xinjianl@andrew.cmu.edu
Carnegie Mellon University

大附 克年
katsutoshi.ohtsuki@microsoft.com
マイクロソフト

中山 英樹
nakayama@ci.i.u-tokyo.ac.jp
東京大学 大学院情報理工学系研究科

1 はじめに

コンピューターやスマートフォンで，日本語や中国語などの非アルファベット言語のテキストを入力するためには，キー系列を目的言語の文字に変換する入力メソッド (Input Method) が必要となる．Input Method において単語系列の確率を評価するための言語モデルとしては，n-gram 言語モデルが一般的に使われている．n-gram 言語モデルで trigram より長い文脈を扱おうとすると，モデルサイズやデータスパースネスの問題からあまり現実ではなく，n-gram とは別のモデルで長い文脈を扱うことが多い．一方で，RNN や LSTM をはじめとするニューラル言語モデルは，長い文脈情報をモデルに含むことができ，n-gram 言語モデルよりも高い精度を達成している [1, 8, 9] だけでなく，扱う文脈を長くしても n-gram 言語モデルのようにモデルサイズが増大していかないという特徴がある．更に，カスタム ASIC [5] という新しい専用ハードウェア処理装置の開発により，ニューラルベースのモデルが今後多くのデバイスに適用されることが期待されている．

通常，ニューラルベースの音声認識や機械翻訳サービスは，モデルサイズや計算リソースの要件からクラウドサービスとして提供されていることが多く，その処理は入力チャンクに対するバッチ処理として実現されている．それに対して Input Method は，キー入力に対して対話的に実行され高速な応答速度が要求されるため，ニューラル言語モデルを適用する際には，既存のバッチ処理に対する高速化手法だけでなく，逐次的な入力に対する高速化を実現する必要がある．

本稿では，ニューラル言語モデルによる Input Method の実時間処理を実現するための高速化について検討する．ニューラル言語モデルと Viterbi デコーディングによる Input Method の概念図を図 1 に示す．多くのニューラルベースのシステムと同様に，実行速度のボトルネックは言語モデル確率を得るための softmax 演算部分となる．Input Method で必要な逐次的な入力に対する softmax 演算を高速化するために，時刻 t におけるサブセット語彙 V_t を構築する逐次語彙選択 (incremental vocabulary selection, IVS) というアプローチを提案する．$|V_t|$ は通常，元の語彙サイズの 1% 未満と小さく，softmax の演算を V_t に限定することで，演算コストを大幅に削減することが期待できる．

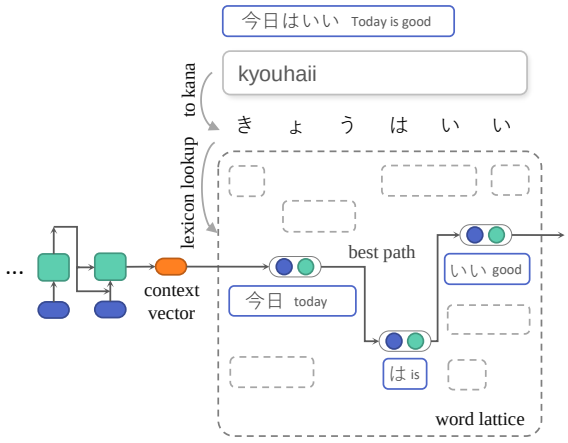

図 1: ニューラル言語モデルと Viterbi デコーディングによる Input Method

2 関連研究

前述したように，ニューラル言語モデルでは softmax 演算が最も計算コストを必要とし，特に語彙サイズが大きい場合に顕著である．Differentiated softmax [2, 4] は，頻度の低い単語の embedding サイズを減らすこと

により演算量を削減している．評価単語がわかっているワードラティスデコーダーでは，構造化された出力層によってすべての語彙に対する確率分布計算を避けることが行われる [10, 6, 11]．単語単位ではなく文字単位で処理を行う character-level RNN [8] は，語彙サイズを大幅に減らすことができるが，漢字を含む日本語や中国語では英語などのラテン文字の言語に比べて削減効果は小さい．

softmax のコストを削減する別のアプローチとして語彙の操作がある．機械翻訳では，原文が与えられると，翻訳前に訳語の語彙を制限することができ，サブセット語彙で softmax を計算することができる [3, 7]．ただし，Input Method の場合，必要な語彙はキー入力によって徐々に増加するために，その動的な変化に対応する必要がある．

3 提案手法

Input Method における逐次的なキー入力に対してニューラル言語モデルの softmax 演算量を削減する逐次語彙選択を提案する．アルゴリズムをアルゴリズム 1 に示す．Π は LSTM 状態の各時刻における最適経路の連想配列である．各時刻での最適経路を得るための辞書 lookup 操作として $[\cdot]$ を定義する．時刻 t までの入力系列 $(x_i, ..., x_t)$ に対するラティス語彙 D_t は式 1 のように計算することができる．

$$D_t = \bigcup_{i=1}^{t} \mathrm{match}(x_i, ..., x_t), \tag{1}$$

ここで，関数 match$(\cdot)$ は，部分入力系列にマッチするすべての単語を返す．たとえば，入力系列「あした」を考えると，D_t には，「あした」，「した」，「た」に一致するすべての単語が含まる．新たな文字 x_t が入力されると，時刻 t までの入力系列に対してすべての可能な出力単語を含む語彙 V_t を次のように構築する．

$$V_t = \bigcup_{i=1}^{t} D_i. \tag{2}$$

仮説 (π_{t-l}, w_t) を評価するために必要な確率 $p(w_t|\pi_{t-l})$ はすでに計算され，π_{t-l} にキャッシュされていることが期待される．しかし，図 2 に示すように，V_t は逐次更新されるので，前の時刻で計算された

アルゴリズム 1 逐次語彙選択

Initialize:
  $B \leftarrow$ beam size
  $\Pi \leftarrow$ hypothesis dictionary
  $\hat{V} \leftarrow$ samples from vocabulary
  $t \leftarrow$ current step t
$V_t \leftarrow$ sub vocabulary with Eq. 2
$V_t \leftarrow V_t \cup \hat{V}$
$V_{\mathrm{fix}} \leftarrow$ empty set
for $k \leftarrow t-1$ to 1 do
  $V_{\mathrm{fix}} \leftarrow V_{\mathrm{fix}} \cup (V_t \setminus V_k)$
  $V_k \leftarrow V_t$
Re-compute the logits on V_{fix} for all past hyps
Evaluate $\Pi[t]$
$\Pi[t] \leftarrow$ best B hyps in $\Pi[t]$
Update LSTM state for $\Pi[t]$
output $\Pi[t]$

softmax 分布は単語 w_t を含まないことがある． $\Pi[t]$ を評価するには，すでに計算された softmax 分布で欠けている語彙項目を修正する必要がある．追加された語彙の logits を分母に追加することにより，時刻 k での古い確率を修正することができる．

$$P(y_k = i|h_k) = \frac{\exp(h_k^\top W_i)}{\sum_{j \in V_k} \exp(h_k^\top W_j) + \sum_{j \in d} \exp(h_k^\top W_j)}. \tag{3}$$

ここで，W は出力層の分散表現行列である．h_k は経路 π_k のキャッシュされた LSTM 状態で，これを使って必要な logit を計算することができる．d は時刻

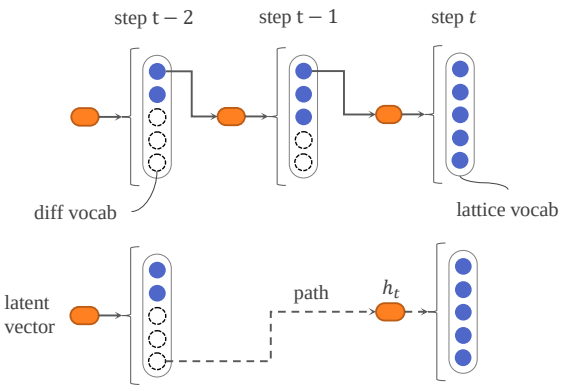

図 2: 時刻によって増加する評価語彙

$V_t \setminus V_k$ の間の語彙の差分である．実際には，経路ごとに不足する語彙が異なるため，必要なすべての語彙の和集合 V_{fix} を計算してから，それらの log をまとめて再計算する．

最初のステップの語彙はかなり小さいので，すべての語彙からサンプリングしたサブセット語彙として $\tilde{V}$ を導入する．機械翻訳におけるビームサーチとは対照的に，異なる数の単語を含む経路を評価する必要がある．$\tilde{V}$ と V_t を使用することによって単語の確率を近似する．

最適経路が決定されると，$\Pi[t]$ の B 経路の LSTM 状態を更新する．最適経路の $\Pi[t]$ が変換結果として返される．

4 実験

4.1 実験設定

BCCWJ コーパスの 5.8M 文を用いて評価実験を行った．データセットの 70% を学習用，20% を開発用，10% を評価用に分割した．データセットに出現する 611k 異なり単語のうち頻度上位 50k 語（被覆率 97.3%）を Input Method の語彙とした．LSTM 言語モデルの学習は，バッチサイズ 384，dropout 確率は 0.9，Adam 最適化の学習率 0.001 で行った．埋め込み層および隠れ層の次元数はそれぞれ 256 次元とした．また，Viterbi デコーディングのビーム幅は 10 とした．

4.2 精度評価

LSTM 言語モデルと n-gram 言語モデルによる Input Method 変換精度を，perplexity と文正解率とで比較した．n-gram 言語モデルは，カットオフ無しで Kneser-Ney 平滑化を適用した．評価結果を表 1 に示す．表 1 をみると，LSTM 言語モデルは trigram 言語モデルに対して，perplexity を 68.11 から 41.39 へと約 40% 削減し，1-best 正解率および 10-best 正解率もそれぞれ約 10% 改善していることが確認できる．

4.3 速度評価

LSTM 言語モデルの逐次語彙選択による実行速度の削減を評価した．本評価では，言語モデルの確率計算に必要な時間を比較した．LSTM 言語モデルでは，LSTM と softmax の演算部分を，n-gram 言語モデルでは，確率の look-up に要する部分を対象としている．ラティス構築などは実装への依存性が高いので比較からは除外した．実験結果は，キーストロークごとの演算時間を示す．また softmax に要した演算時間もあわせて示す．

model	pp	top1 hit %	top10 hit %
uni-gram	833.55	26.95	45.85
bi-gram KN	99.30	51.15	78.10
tri-gram KN	68.11	55.60	79.65
LSTM baseline	41.39	61.20	88.30

表 1: 変換精度評価結果

表 2 に評価結果を示す．提案手法である逐次語彙選択 (IVS) が，ベースラインである LSTM (LSTM baseline w/ batch) の 84 倍の演算速度を実現していることが確認できる．フレーム当たりのアクティブな語彙は，50k の語彙サイズに対して，たかだか数百単語であった．IVS がキーストロークごとに必要な演算時間は 3ms であり，演算性能の低いデバイスであっても実時間処理が可能なレベルである．IVS により softmax に要する演算時間が大幅に削減され，必要な演算はほとんど LSTM 自体のものとなっている．

IVS に，いくつかのサンプリング手法を適用した評価も行った．サンプル数 400 で top sampling と uniform sampling を適用した場合，top sampling の方が改善精度が高かった．

Softmax の演算量を削減する方法として Differentiated softmax があるが，本評価のように語彙数が大きい場合には，その削減効果は限定的であった (D-Softmax[2], D-Softmax*[4])．語彙数を抑制する方法として，文字あるいは subword ベースの LSTM がある．隠れ層と埋め込み層をそれぞれ 1024 次元，512 次元とした文字ベースの LSTM（語彙数 3,717）の評価を行った (Char LSTM) が，入力メソッドタスクでは文字ベースのアプローチは曖昧性が高く，ビームサイズを 50 まで大きくしても十分な精度を得ることができなかった (Char LSTM w/ large beam).

5 おわりに

Input Method は，日本語や中国語などの入力においてなくてはならないものであり，またキー入力および操作に対して遅延があると操作性を大きく損なって

*1 ビームサイズは 50 に設定されている．

model	total time (ms)	softmax time (ms)	top-1 hit (%)	top-10 hit (%)
tri-gram Kneser-Ney	0.0025	-	55.6	79.6
LSTM baseline w/o batch	526	513	61.2	88.3
LSTM baseline w/ batch	87	84	61.2	88.3
Char LSTM	63	21	53.2	79.8
Char LSTM w/ large beam *1	152	55	54.4	85.2
D-Softmax	38	35	60.8	86.8
D-Softmax*	37	35	60.9	87.5
IVS	3	1	58.8	86.9
w/ top sampling	4	2	60.1	88.2
w/ uniform sampling	4	2	58.9	87.2
w/ self-norm	3	1	60.1	88.2
non-incremental VS	11	2	58.8	86.9

表 2: 処理速度評価結果

しまう．本稿では，n-gram 言語モデルより精度の高い LSTM 言語モデルを日本語の入力メソッドに適用することを検討した．LSTM 言語モデルは，softmax の演算が処理時間のボトルネックとなるが，提案した逐次語彙選択 (IVS) により，ベースラインの LSTM 言語モデルと同程度の精度を維持しながら，1 フレーム当たりの言語モデル演算時間を 3ms にまで削減することができた．これは，比較的低スペックのデバイスでも遅延を感じずに入力メソッドを使うことができる実用可能なレベルの速度であるといえる．

参考文献

[1] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. Journal of machine learning research, 3(Feb):1137–1155, 2003.

[2] Wenlin Chen, David Grangier, and Michael Auli. Strategies for training large vocabulary neural language models. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), volume 1, pages 1975–1985, 2016.

[3] Sebastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. In 53rd Annual Meeting of the Association for Computational Linguistics, ACL-IJCNLP 2015, pages 1–10. Association for Computational Linguistics (ACL), 2015.

[4] Armand Joulin, Moustapha Cissé, David Grangier, Hervé Jégou, et al. Efficient softmax approximation for gpus. In International Conference on Machine Learning, pages 1302–1310, 2017.

[5] Norman P. Jouppi and Cliff Young. In-datacenter performance analysis of a tensor processing unit. 2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA), pages 1–12, 2017.

[6] Hai Son Le, Ilya Oparin, Alexandre Allauzen, Jean-Luc Gauvain, and FranÃ§ois Yvon. Structured output layer neural network language model. 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 5524–5527, 2011.

[7] Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. Vocabulary manipulation for neural machine translation. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), volume 2, pages 124–129, 2016.

[8] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In Eleventh Annual Conference of the International Speech Communication Association, 2010.

[9] Tomas Mikolov and Geoffrey Zweig. Context dependent recurrent neural network language model. 2012 IEEE Spoken Language Technology Workshop (SLT), pages 234–239, 2012.

[10] Andriy Mnih and Geoffrey E Hinton. A scalable hierarchical distributed language model. In Advances in neural information processing systems, pages 1081–1088, 2009.

[11] Yongzhe Shi, Weiqiang Zhang, Jia Liu, and Michael T. Johnson. Rnn language model with word clustering and class-based output layer. EURASIP J. Audio, Speech and Music Processing, 2013:22, 2013.