中間層の利用によるRNN言語モデルの表現力向上

高瀬 翔　鈴木 潤　永田 昌明
NTT コミュニケション科学基礎研究所

{takase.sho, suzuki.jun, nagata.masaaki}@lab.ntt.co.jp

1 はじめに

ニューラルネットワークを用いた言語モデル（ニューラル言語モデル）は，近年の自然言語処理分野の発展における中核的な技術である．例えば，翻訳 [7] や要約 [13] などの自然言語生成分野で成功を収めている，ニューラルネットワークを用いたエンコーダ・デコーダモデルは，条件付きのニューラル言語モデルと解釈できる．また，Skip-gram [12] などの単語分散表現の学習手法は，大規模な語彙およびデータを扱うためのニューラル言語モデルとみなせる．すなわち，ニューラル言語モデルの性能向上は，自然言語処理分野の様々なタスクに貢献すると考えられる．本研究では，ニューラル言語モデル，特に，Recurrent Neural Network（RNN）言語モデルの性能向上に取り組む．

言語モデルでは，同時確率を条件付き確率の積によって計算する．具体的には，長さ T の系列 $w_1, ..., w_T$（以下，これを $w_{1:T}$ と表す）の同時確率 $P(w_{1:T})$ を以下の式で計算する．

$$P(w_{1:T}) = P(w_1) \prod_{t=1}^{T-1} P(w_{t+1}|w_{1:t}). \quad (1)$$

なお，一般的に $P(w_1)$ は 1 と仮定し，この項の計算は行わない（詳細は Zaremba ら [18] の実装を参照のこと[1]）．RNN 言語モデルでは，$w_{1:t}$ を固定長のベクトルで表現し，さらに，このベクトルに変換行列，およびソフトマックス関数を適用して得られた，語彙に対する確率分布から条件付き確率 $P(w_{t+1}|w_{1:t})$ を得る．

RNN 言語モデルは様々な正則化手法を組み合わせることで，極めて高い性能を達成できることが知られている [9, 10]．一方で，言語モデルを行列分解の問題として解釈した際に，既存の RNN 言語モデルは出力する行列のランクが小さく，表現力が低いことが Yang らにより指摘されている [17]．さらに，Yang らは系列を表す固定長のベクトルから複数の確率分布を計算し，組み合わせることで，出力する行列のランクが上がること，および性能が向上することを示した．本稿では，Yang らの手法の一般化として，複数層の RNN について，入力である単語の分散表現も含めた，全ての層から確率分布を計算する手法を提案する．本手法は中間層から出力への短縮路となっており，ニューラルネットワークの学習の観点からは，誤差逆伝播における勾配消失問題の解消も期待できる [15]．

実験を通し，提案手法は言語モデルの標準的なベンチマークデータである，Penn Treebank [8]，WikiText-2 [11] において，最高性能を達成することを示す．加えて，要約タスクの一種である，ヘッドライン生成タスク [13] における有効性を示す．

2 RNN 言語モデル

RNN 言語モデルの概要を述べる．単語の語彙数を V とし，時刻 t において単語が出現する確率分布を $P_t \in \mathbb{R}^V$ とする．時刻 t での入力単語 w_t の one-hot 表現を $x_t \in \{0,1\}^V$ としたとき，N 層の RNN を用いた言語モデルは，次の時刻の単語の確率分布 P_{t+1} を以下の式で得る．

$$P_{t+1} = \text{softmax}(W h_t^N), \quad (2)$$
$$h_t^n = f(h_t^{n-1}, h_{t-1}^n), \quad (3)$$
$$h_t^0 = E x_t. \quad (4)$$

ここで，n 層目の RNN の隠れ状態の次元数を D_{h^n}，単語の分散表現の次元数を D_e とすると，$W \in \mathbb{R}^{V \times D_{h^N}}$ は重み行列[2]，$E \in \mathbb{R}^{D_e \times V}$ は単語の分散表現を記した行列となる．なお，$D_{h^0} = D_e$ とする．また，各層において，時刻 $t = 0$ における隠れ状態はゼロベクトルとする，すなわち，$h_0^n = \mathbf{0}$ である．式 3 における，$f(\cdot)$ は LSTM [3] や Recurrent Highway Network [20] など，任意の RNN を表す関数である．本研究では，高い性能を達成している Merity らのモデルと同様，3 層の LSTM を用いる [10]．

3 言語モデルの行列分解的解釈

Yang らは言語モデルの学習は行列分解とみなすことができると指摘した [17]，その概観を述べる．単語列 $w_{1:t}$ を文脈と呼び，c_t で表すと，自然言語は文脈と条件付き確率分布のペアの集合 $\mathcal{L} = \{(c_1, P^*(X|c_1)), ..., (c_U, P^*(X|c_U))\}$ と考えられる．ここで，U は存在可能な文脈の数，$X \in \{0,1\}^V$ は単語の one-hot 表現を表す変数である．

さらに，行列の各行に softmax 関数を適用する関数を Softmax としたとき，この関数を適用することで真の分布を得られる行列 $A \in \mathbb{R}^{U \times V}$，および，各文脈 c_t に対する，RNN の最終層の隠れ状態 $h_{c_t}^N$ を縦に

[1] https://github.com/wojzaremba/lstm

[2] 正確には，重み行列に加え，バイアス項を適用するが，議論を簡単にするため，本稿ではバイアス項を省略する．

並べた行列 $H \in \mathbb{R}^{U \times D_{h^N}}$ を考える．すなわち，

$$\mathrm{Softmax}(A) = \begin{bmatrix} P^*(X|c_1) \\ P^*(X|c_2) \\ \cdots \\ P^*(X|c_U) \end{bmatrix}; H = \begin{bmatrix} h^N_{c_1} \\ h^N_{c_2} \\ \cdots \\ h^N_{c_U} \end{bmatrix}. \quad (5)$$

このとき，式 2 および式 5 から，言語モデルの学習は A，H，および行列 W について，以下の式を満たすパラメータを得ることであると言える．

$$HW^{\mathrm{T}} = A \quad (6)$$

式 6 から，言語モデルの学習は行列分解とみなすことができる．また，一般的に，RNN 言語モデルでは D_{h^N} は語彙数 V や文脈数 U より小さいため，モデルの出力する行列のランクは，D_{h^N} となる．このため，$\mathrm{rank}(A)$ に比べて D_{h^N} が小さな値である場合，RNN 言語モデルは真の分布を表現できない．

Yang らはさらに，1．自然言語は文脈依存性が高く，また，文脈も多様であるため，任意の文脈に対する条件付き確率分布を表現可能な基底は考えづらい，すなわち，U を圧縮することは難しいこと，2．単語の意味も多様であるため，和差算により任意の単語を表現可能な基底も考えづらい，すなわち，V を圧縮することも難しいと仮定した．このことから，$\mathrm{rank}(A)$ は語彙数程度（すなわち，$10^4 < \mathrm{rank}(A)$）には大きく，たかだか 10^2 規模の D_{h^N} は A を表現するには小さすぎることが指摘されている．

4 提案手法：Direct Output Connection

ランクの大きな行列を表現するため，Yang らは，RNN の最終層の隠れ状態 h^N から複数個の確率分布を計算し，その重み付き平均を最終的な確率分布とする手法を提案した．本節では，Yang らの手法を一般化した手法として，中間層からも確率分布を計算する，すなわち，中間層を最終的な出力に接続する手法（Direct Output Connection，以下 DOC と呼ぶ）を提案する．

DOC では，時刻 $t+1$ の確率分布について，式 2 に変わり，以下の式で確率分布を計算する．

$$P_{t+1} = \sum_{s=1}^{S} \pi_{s,w_{1:t}} \ \mathrm{softmax}(\tilde{W} k_{s,w_{1:t}}), \quad (7)$$

$$s.t. \sum_{s=1}^{S} \pi_{s,w_{1:t}} = 1 \quad (8)$$

ここで，$\pi_{s,w_{1:t}}$ は各確率分布に対する重みであり，$k_{s,w_{1:t}} \in \mathbb{R}^d$ は RNN 言語モデルの各層の隠れ状態 h^n から計算した固定長ベクトル，$\tilde{W} \in \mathbb{R}^{V \times d}$ は重み行列である．すなわち，S 個の確率分布の重み付き平均により，時刻 $t+1$ での単語の確率分布を計算する．文脈 c に対する重み $\pi_{s,c}$ を要素とする $U \times U$ の対角行列を Φ としたとき，式 7 から，式 6 は以下のように書き換えられる．

$$\log \sum_{s=1}^{S} \Phi \ \mathrm{Softmax}(K_s \tilde{W}^{\mathrm{T}}) = A \quad (9)$$

表 1: Penn Treebank および WikiText-2 の詳細．

		Penn Treebank	WikiText-2
語彙		10,000	33,278
トークン数	Train	929,590	2,088,628
	Valid	73,761	217,646
	Test	82,431	245,569

表 2: 提案手法を学習する際のハイパーパラメータ．

ハイパーパラメータ	Penn Treebank	WikiText-2
学習率	20	15
バッチサイズ	12	15
D_e	280	300
D_{h^1}	960	1150
D_{h^2}	960	1150
D_{h^3}	620	650
x_t へのドロップアウト率	0.1	0.1
h^0_t へのドロップアウト率	0.4	0.65
h^1_t, h^2_t へのドロップアウト率	0.225	0.2
h^3_t へのドロップアウト率	0.4	0.4
$k_{s,w_{1:t}}$ へのドロップアウト率	0.6	0.6

ここで，$K_s \in \mathbb{R}^{U \times d}$ は各文脈 c に対するベクトル $k_{s,c}$ を縦に並べた行列である．式 9 の左辺は行列演算に加え，非線形変換が適用されており，任意のランクを取ることができる．これにより，モデルの出力する行列は次元数 d に制限されず，真の分布と同一のランクを取ることができる．

重み $\pi_{s,w_{1:t}}$，および，$k_{s,w_{1:t}}$ の計算法について記す．重み $\pi_{s,w_{1:t}}$ を各次元の要素とするベクトル $\pi_{w_{1:t}} \in \mathbb{R}^S$ は RNN 言語モデルの最終層の隠れ状態，および，重み行列 $W_\pi \in \mathbb{R}^{S \times D_{h^N}}$ から，以下の式により計算する．

$$\pi_{w_{1:t}} = \mathrm{softmax}(W_\pi h^N_{1:t}) \quad (10)$$

$k_{s,w_{1:t}}$ は RNN 言語モデルの n 層目の隠れ状態 h^n_t から，以下の式で計算する．

$$k_{s,w_{1:t}} = W_s h^n_t \quad (11)$$

ここで，$W_s \in \mathbb{R}^{d \times D_{h^n}}$ は重み行列であり，h^n_t から計算される $k_{s,w_{1:t}}$ の数を i_n とすると，$\sum_{n=0}^{N} i_n = 1$ である．すなわち，DOC では，単語の分散表現（h^0_t）を含めた任意の中間層から合計 S 個の確率分布を計算する．なお，$i_N = S$ のとき，DOC は Yang らの手法に一致する．また，提案手法は中間層を直接出力に接続しているため，誤差逆伝播における勾配消失の解消，および，正則化も期待できる [15]．

しかしながら，このように複数の出力の重み付き平均を取る手法では，学習初期において性能が良い出力に大きな重みが付与され続けてしまい，不均衡なまま収束してしまう問題が知られている [14]．実際，DOC でも，浅い層の出力に高い重みが付与され続ける現象が観測された．これを防ぐために，本研究では，Shazeer ら [14] と同様，式 10 のミニバッチ毎の変動係数を正則化項として加える．言い換えれば，各ミニバッチにおいて，各出力への重みの合計値が同程度になるような正則化を加える．具体的には，$w_b, w_{b+1}, ..., w_{\tilde{b}}$ のミニバッチについて，以下の式を計算する．

$$B = \sum_{t=b}^{\tilde{b}} \pi_{w_b} \quad (12)$$

$$\beta = \left(\frac{\mathrm{std}(B)}{\mathrm{avg}(B)} \right)^2 \quad (13)$$

表 3: Penn Treebank データセットにおける各手法での Perplexity．上段は既存研究で報告されている値．下段は $S = 20$ において，ハイパーパラメータを変化させた際のベースラインおよび提案手法の性能．

Model		#Param	#DOC i_3	i_2	i_1	i_0	Validation	Test
LSTM (medium) [18]		20M	1	0	0	0	86.2	82.7
LSTM (large) [18]		66M	1	0	0	0	82.2	78.4
Variational LSTM (medium) [1]		20M	1	0	0	0	81.9 ± 0.2	79.7 ± 0.1
Variational LSTM (large) [1]		66M	1	0	0	0	77.9 ± 0.3	75.2 ± 0.2
Variational RHN [20]		32M	1	0	0	0	71.2	68.5
Variational RHN + WT + IOG [16]		29M	1	0	0	0	67.0	64.4
2-layer skip connections LSTM [9]		24M	1	0	0	0	60.9	58.3
AWD-LSTM [10]		24M	1	0	0	0	60.0	57.3
AWD-LSTM-MoS [17]		22M	15	0	0	0	56.54	54.44
$S = 20$ の際のベースラインおよび提案手法	AWD-LSTM-MoS	22M	20	0	0	0	56.88	54.79
	AWD-LSTM-DOC	22M	15	0	0	5	56.21	54.28
	AWD-LSTM-DOC	23M	15	0	5	0	55.26	53.52
	AWD-LSTM-DOC	23M	15	5	0	0	54.87	53.15
	AWD-LSTM-DOC (λ_β=0.001)	23M	15	5	0	0	**54.62**	**52.87**
	AWD-LSTM-DOC	22M	10	5	0	5	56.46	54.18
	AWD-LSTM-DOC	23M	10	5	5	0	56.00	54.37

表 4: WikiText-2 における各手法での Perplexity．

Model	#Param	#DOC i_3	i_2	i_1	i_0	Validation	Test
Variational LSTM + IOG [16]	70M	1	0	0	0	95.9	91.0
2-layer skip connection LSTM [9]	24M	1	0	0	0	69.1	65.9
AWD-LSTM [10]	33M	1	0	0	0	68.6	65.8
AWD-LSTM-MoS [17]	35M	15	0	0	0	63.88	61.45
提案手法：AWD-LSTM-DOC (λ_β=0.001)	37M	15	5	0	0	**60.97**	**58.55**

表 5: Penn Treebank での対数尤度行列のランク．

Method	Valid	Test
AWD-LSTM [10] ($D_{h^3} = 400$)	401	401
AWD-LSTM-MoS ($i_3 = 20$)	10000	10000
AWD-LSTM-DOC ($i_3 = 15, i_2 = 5$)	10000	10000

ここで，std(·)，avg(·) はそれぞれ入力の標準偏差と平均値を返す関数である．学習の際には，β に重み係数 λ_β をかけ合わせた項を損失関数に加える．

5 実験

提案手法の言語モデルタスクにおける性能を明らかにする．また，言語モデルの応用タスクである，自然言語生成における性能を検証するため，生成的要約タスクである，ヘッドライン生成タスクでの実験も行う．

5.1 実験設定

言語モデル 言語モデルのベンチマークデータとして，多くの研究で利用されている，Penn Treebank データセット[3] [8]，および，WikiText-2 データセット[4] [11] を用いて実験を行う．既存研究との公正な比較を行うため，公開されている，前処理済みのデータを用いる．データセットの語彙数，トークン数については表 1 に記したとおりである．Yang らと同様，Merity らの 3 層 LSTM 言語モデル[5][10] を元に実装を行った．ハイパーパラメータについても，最終層 $k_{s,w_{1:t}}$ へのドロップアウト率以外，Yang らと同様の値を用いた．最終層へのドロップアウト率については，式 13 の β への影響が大きかったため，0.3 から 0.6 まで 0.1 刻みで探索し，最も良い値であった 0.6 を採用した．具体的には表 2 に記したとおりである．

[3]http://www.fit.vutbr.cz/~imikolov/rnnlm/

[4]https://einstein.ai/research/the-wikitext-long-term-dependency-language-modeling-dataset

[5]https://github.com/salesforce/awd-lstm-lm

ヘッドライン生成 ニュース記事の一文目からヘッドラインを生成する，ヘッドライン生成タスクでの実験を行う．具体的には，Rush ら [13] の構築した，約 400 万の文–ヘッドライン対からエンコーダ・デコーダモデルを学習し，性能を検証する．エンコーダ・デコーダについては，Luong らのモデル [7] をベースとし[6]，ハイパーパラメータや語彙数などの実験設定については，学習アルゴリズムとして Adam [4] ではなく SGD を用いた点，Gradient Clipping の値を 1 にした点以外は Kiyono ら [5] と同一とした．

評価について，Rush らの公開しているテストデータは，1 単語のみの入力文からのヘッドライン生成など，問題として破綻している事例が含まれると，Zhou らにより指摘されている [19]．このため，本研究では，これを解決した，Zhou らのテストデータ，および，Kiyono らのテストデータを用いて評価を行う．これらのデータは，それぞれ，2000，10000 の文–ヘッドライン対を含んでいる．

5.2 結果

言語モデル 提案手法の Penn Treebank，WikiText-2 での Perplexity をそれぞれ表 3，4 の下部に示す．各表の上部には，既存研究で報告されている Perplexity の値も比較のために記してある．なお，訓練データから学習したモデルの質のみ議論を絞るため，テスト時に，テストデータにおける観測済みの系列の統計量を用いる手法 [2, 6] は除外した．また，表 3 の下部には，$S = 20$，すなわち，確率分布の総数を 20 とした際の，ベースラインである Yang らの手法（AWD-LSTM-MoS）[7] [17]，および，提案手法における各層

[6]アテンション機構を含むエンコーダ・デコーダであり，アテンション計算後の層を最終層（h^N）とみなす．

[7]実験には公開されている，以下の実装を用い，i_3 以外のハイパーパラメータは彼らと同一にした．https://github.com/zihangdai/mos

表 6: ヘッドライン生成タスクでの各 ROUGE の F1 値.

Method	#DOC i_3	#DOC i_2	Test (Zhou) ROUGE-1	Test (Zhou) ROUGE-2	Test (Zhou) ROUGE-L	Test (Kiyono) ROUGE-1	Test (Kiyono) ROUGE-2	Test (Kiyono) ROUGE-L
EncDec	1	0	45.92 ± 0.11	24.16 ± 0.10	42.70 ± 0.07	45.86 ± 0.03	23.70 ± 0.08	42.80 ± 0.04
EncDec+MoS	2	0	46.04 ± 0.24	24.26 ± 0.25	42.96 ± 0.23	46.11 ± 0.19	23.85 ± 0.18	43.02 ± 0.16
EncDec+DOC	2	2	**46.40 ± 0.24**	**24.36 ± 0.15**	**43.18 ± 0.20**	**46.13 ± 0.09**	**23.85 ± 0.04**	**43.03 ± 0.10**

からの確率分布の数を変化させたときの性能を記している．ここで，$\lambda_\beta = 0.001$ と表記のある行以外は，$\lambda_\beta = 0$ である．

表 3 から，提案手法は既存研究よりも低い Perplexity，すなわち，高い性能を達成可能であることが分かる．RNN の最終層から出力する確率分布の数のみを増やした場合（AWD-LSTM-MoS [17] の $i_3 = 20$）は性能が悪化してしまっていることに比べ，提案手法では性能が向上していることから，RNN の中間層を出力とつなげることにより，質の良いネットワークが学習できていると考えられる．一方で，i_0 や i_1 のような浅い層からの出力を組み合わせるよりも，i_2 と i_3 の出力のみを用いた方が高い性能となっている．これは，質の高い確率分布を出力するためには，RNN の層を一定数積み重ねる必要があることを示唆している．実際，既存研究では 2 層 LSTM を用いるのが主流であり [18, 1]，ハイパーパラメータを調整することにより，高い性能が報告されている [9]．提案手法である DOC は λ_β での正則化を用いることでさらに性能を向上させることができ，最終的に，既存の最高性能のモデルに対し，Penn Treebank では 1.5，WikiText-2 では 3.0 程度，Perplexity を下げることができた．

Penn Treebank における，学習済みモデルの出力した対数尤度行列（式 6，式 9）のランクを表 5 に示す．3 層 LSTM 言語モデル [10] はランクが最終層の次元数に制限されているのに対し[8]，提案手法，および，Yang らの手法は，語彙数と同一のランクを達成している．このことから，複数の確率分布を組み合わせることにより，ランクの観点において，真の分布と同一の行列を出力できることが分かる．

ヘッドライン生成 Zhou らのテストデータ（Test (Zhou)），および，Kiyono らのテストデータ（Test (Kiyono)）における，ベースライン，Yang らの手法，提案手法の F 値ベースの ROUGE 値を表 6 に示す．ここで，ベースラインである，アテンション機構付きのエンコーダ・デコーダモデルを EncDec，最終層のみから複数個確率分布を計算する Yang らの手法を EncDec+MoS，提案手法である，中間層も用いた手法を EncDec+DOC と表す．なお，初期値の差による誤差を低減するため，各設定について，乱数のシードを変えて学習を行ったモデルを 3 つ用意し，その平均と標準偏差を記した．表 6 から，提案手法は両データにおいて，ベースライン，および，Yang らの手法よりも高い性能を達成していることが分かる．Test (Kiyono) では，Yang らの手法と提案手法とはほとんど同じスコアとなっているが，標準偏差に関しては，提案手法の方が小さい．この結果は，提案手法では，中間層も出力に接続することで，性能を向上させるだけでなく，頑健なモデルの構築も可能であることを示唆している．

[8]厳密には，バイアス項により基底が増えることがあり，表 5 のように，次元数 +1 がランクの最大値となる．

6 おわりに

本稿では，RNN 言語モデルについて，任意の中間層から確率分布を出力し，重み付き平均を取ることで，表現力を向上させる手法を提案した．言語モデルの標準的なベンチマークデータである，Penn Treebank と WikiText-2 において，テストデータの情報をテスト時に利用する手法を除き，最高性能を達成した．また，ヘッドライン生成タスクにおいても実験を行うことにより，提案手法はエンコーダ・デコーダの性能向上にも寄与することを示した．

参考文献

[1] Yarin Gal and Zoubin Ghahramani. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. In *NIPS*, 2016.
[2] Edouard Grave, Armand Joulin, and Nicolas Usunier. Improving Neural Language Models with a Continuous Cache. In *ICLR*, 2017.
[3] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, Vol. 9, No. 8, pp. 1735–1780, 1997.
[4] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015.
[5] Shun Kiyono, Sho Takase, Jun Suzuki, Naoaki Okazaki, Kentaro Inui, and Masaaki Nagata. Source-side prediction for neural headline generation. *CoRR*, 2017.
[6] Ben Krause, Emmanuel Kahembwe, Iain Murray, and Steve Renals. Dynamic evaluation of neural sequence models. *CoRR*, 2017.
[7] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *EMNLP*, pp. 1412–1421, 2015.
[8] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, Vol. 19, No. 2, pp. 313–330, 1993.
[9] Gábor Melis, Chris Dyer, and Phil Blunsom. On the state of the art of evaluation in neural language models. *CoRR*, 2017.
[10] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and Optimizing LSTM Language Models. *CoRR*, 2017.
[11] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer Sentinel Mixture Models. In *ICLR*, 2017.
[12] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*, pp. 3111–3119. 2013.
[13] Alexander M. Rush, Sumit Chopra, and Jason Weston. A Neural Attention Model for Abstractive Sentence Summarization. In *EMNLP*, pp. 379–389, 2015.
[14] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *ICLR*, 2017.
[15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.
[16] Sho Takase, Jun Suzuki, and Masaaki Nagata. Input-to-output gate to improve rnn language models. In *IJCNLP*, pp. 43–48, 2017.
[17] Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. Breaking the softmax bottleneck: A high-rank RNN language model. *CoRR*, 2017.
[18] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent Neural Network Regularization. In *ICLR*, 2014.
[19] Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. Selective encoding for abstractive sentence summarization. In *ACL*, pp. 1095–1104, 2017.
[20] Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber. Recurrent Highway Networks. *ICML*, pp. 4189–4198, 2017.