ニューラル機械翻訳のためのアテンション確率のスムージングとゲーティング学習

張 瀟廬
愛媛大学
zhang@ai.cs.ehime-u.ac.jp

二宮 崇
愛媛大学
ninomiya@cs.ehime-u.ac.jp

田村 晃裕
同志社大学
aktamura@mail.doshisha.ac.jp

1 はじめに

近年、人工知能の発展とともに、自然言語処理分野においてもニューラルネットワーク (Neural Network) に基づく手法が主流になっており、特に機械翻訳のタスクにおいてはニューラルネットワークを用いた機械翻訳 (ニューラル機械翻訳) が高い精度と自然な翻訳を実現することから大きく注目を集めている。ニューラル機械翻訳モデルとして、LSTM (Long-Short Term Memory) を用いた回帰型ニューラルネットワーク (Recurrent Neural Network; RNN) に基づくエンコーダー・デコーダーモデルが提案され [1]、さらにアテンション機構 (Attention Mechanism)[2] によって翻訳精度が大きく改善された。Transformer[3] は、回帰的構造を持たずアテンション機構だけから構成されるエンコーダー・デコーダーモデルであり、言語間アテンションに加え、原言語文や目的言語文における各単語間のアテンション (自己アテンション) を計算することを特徴とする。回帰的構造を持たない代わりに位置エンコーディングを埋め込むことで、単語の位置情報や前後関係を捉えている。また、マルチヘッドをアテンション機構に導入しており、中間表現ベクトルを各ヘッドに分割し、各ヘッドにおいてアテンションを計算することにより様々な文の特徴を捉えることを可能としている。

Transformer は、非常に高い翻訳精度を実現することから、現在も多くの翻訳モデルのベースとなっており、時系列をより強い関係で結びつける Monotonic Attention[4] や、異なるアテンションマスクを導入した Mixed Multi-Head Self Attention[5]、Highway Transformer[6]、Capsule Networks[7]、Multi-Hop Attention[8] などが提案されている。また、アテンションに関する解析では、アテンションの確率計算は単なる重みの計算ではないという報告がなされている [9]。Transformer およびこれらの Transformer に基づく手法のいずれも、そのアテンション計算はソフトマックス関数を使った関連性計算を行っており、計算結果はある単語に対する文中の各単語との関連性を表す離散的な確率分布になっている。ソフトマックス関数はその性質から一つや二つの単語だけにきわめて高いピークの確率を与えることが多いが、そのため、単語確率分布におけるデータスパースネスや過学習の問題が混在している可能性がある。

そこで、本論文ではアテンションの関連性計算に対してスムージング (Smoothing) を適用し、その確率値を調整する手法を提案する。具体的には、ラベルスムージング手法 [10] を用いたアテンションスムージング (Attention Smoothing) と、スムージングのパラメータをモデルに学習させるゲートスムージング (Gate Smoothing) を提案する。WAT の ASPEC 英日翻訳タスクにおいて実験を行い、アテンションスムージングによって BLEU(%) 値が+0.74 向上し、ゲートスムージングによって BLEU(%) 値が+0.99 向上することを確認した。

2 Transformer

Transformer は近年ニューラル機械翻訳で最もよく使われるエンコーダー・デコーダーモデルの一種である。原言語の文をエンコーダーに入力し、エンコーダからは入力に対する中間表現が出力される。デコーダーには、エンコーダーの中間表現および現時点までの訳文を入力し、それらを組み合わせて次

This work is licensed by the author(s) under CC BY 4.0
(https://creativecommons.org/licenses/by/4.0/).

の時刻の単語を生成する。長さ S の原言語の文を $X = \{x_1, x_2, \cdots, x_S\}$ とし、長さ T の目的言語の文を $Y = \{y_1, y_2, \cdots, y_T\}$ とすると、生成の過程は以下の式で表せる。

$$P(y_t|y_{<t}, X) = \mathbb{M}(y_{<t}, X) \tag{1}$$

ただし、$\mathbb{M}$ はモデルを表す。モデルは時刻ごとに次の単語を生成する確率を算出する。

Transformer のエンコーダーおよびデコーダーは、それぞれスタックされた 6 つのレイヤーから構成される。エンコーダー側のレイヤーは自己アテンション層とフィードフォワード層で構成され、デコーダー側のレイヤーは自己アテンション層と言語間アテンション層、フィードフォワード層で構成される。

アテンションの計算について説明する。入力をクエリー q、キー k、バリュー v とする。クエリーとキーで単語間の関連度を計算し、それを重みとしてバリューの重み付き平均値を計算する。計算過程は以下の式で表せる。

$$score = \mathbf{W}^q(q) \cdot \mathbf{W}^k(k)^\top \tag{2}$$

$$a = softmax(score) \tag{3}$$

$$o_i = \sum_{j=1}^{V} a_{ij} \mathbf{W}^v_{j\cdot}(v) \tag{4}$$

ただし、$W^{(q)} \in \mathbb{R}^{Q \times d}, W^{(k)} \in \mathbb{R}^{K \times d}, W^{(v)} \in \mathbb{R}^{V \times d}, a \in \mathbb{R}^{Q \times K}, o_i \in \mathbb{R}^d$、$d$ は隠れ層の次元数、Q, K, V はクエリー、キー、バリューの長さを表し、$i \in \{1, 2, \cdots, Q\}, j \in \{1, 2, \cdots, K\}$ かつ $K = V$ を満たしている。$softmax(\cdot)$ はスコアを正規化するための活性化関数であるソフトマックス関数である。ソフトマックスの計算結果は $\sum_{j=0}^{V} a_{ij} = 1(\forall i)$ という制約が課せられる。

自己アテンションでは、クエリー、キー、バリューはそれぞれ前の層からの出力となる。前の層からの出力は、エンコーダー側では原言語の中間表現であり、デコーダー側では目的言語の中間表現である。言語間アテンションでは、クエリーはデコーダー側の前の層の出力、キーとバリューはエンコーダー側の最終レイヤーの出力になる。

自己アテンションと並んで Transformer において重要な部分はマルチヘッドアテンションである。マルチヘッドの場合は、次元数をヘッド数で割り、ヘッドごとにアテンションの計算を行う。マルチヘッドアテンションは自己アテンションや言語間アテンションに適用される。

3 提案手法

従来のマルチヘッドアテンションは単語間の関連性 a を計算するためにソフトマックス関数を使い、$a_{ij} \in (0, 1)\ s.t. \sum_{j=0}^{V} a_{ij} = 1(\forall i)$ という制約の下で単語間の関連性を表す。ソフトマックス関数で正規化する際には指数関数が適用されるため、関連の高い単語に対する値がより顕著になるような確率値が算出される。その結果、一つや二つの単語にだけきわめて高い確率が割り振られる $(\max_j[a_{ij}] \gg a_{ik}(k \neq j, \forall i))$ ため、アテンション確率においても過学習の問題が混在している可能性が高い。そこで、本研究ではアテンションで単語間の関連性を算出する際にスムージングを行う手法を提案する。

3.1 アテンションスムージング

分類モデルの学習時に損失を計算する際、正解ラベルのみに基づいて損失を計算すると過学習を起こしてしまう可能性がある。ラベルスムージングは、そのような過学習を抑えるための手法であり、正解ラベルに対する確率を一定率低くし、正解ラベル以外のラベルに対する確率を一定率高めることで、ラベル確率分布を滑らかにする。

本研究では、ラベルスムージングの手法をアテンションの算出時に適用するアテンションスムージングを提案する。アテンションスムージングは、ラベルスムージングに倣い、アテンションの計算による単語間関連度を表す確率分布を滑らかにする。具体的には、次式のように、アテンションの確率分布に対して、最も高い確率を一定率 (s) 低くし、その他の確率を一定率 $(1/s)$ 高める方法である。

$$\hat{a}_{ij} = \begin{cases} a_{ij} \cdot s & j = \arg\max_j[a_{ij}] \\ a_{ij} \cdot \frac{1}{s} & j \neq \arg\max_j[a_{ij}] \end{cases} \tag{5}$$

$$o_i = \sum_{j=1}^{V} \hat{a}_{ij} \mathbf{W}^v_{j\cdot}(v) \tag{6}$$

ただし、$s \in \mathbb{R}$ はスムージングの強さを制御するハイパーパラメータであり、$0 < s \leq 1$ を満たしている。s を小さくするほど、過度なピークの情報を抑えてアテンションの単語確率分布を滑らかにする。$s = 1$ の場合、アテンションスムージングを用いない従来手法と同等になる。

This work is licensed by the author(s) under CC BY 4.0
(https://creativecommons.org/licenses/by/4.0/).

表 1　実験結果

モデル	英日 (BLEU(%))
ベースライン	33.65
ベースライン (ダミーパラメータ)	34.03
アテンションスムージング	34.39 (+0.74)
ゲートスムージング	34.64 (+0.99)

3.2　ゲートスムージング

アテンションスムージングでは、スムージングの強さを表すハイパーパラメータ s を人手で調整する必要がある。本研究では、ゲーティングの仕組みを取り入れて、スムージングの強さもパラメータとして学習するゲートアテンションを提案する。ゲートアテンションを数式で表現すると以下の通りとなる。

$$score_s = \mathbf{W}^{sq}(q) \cdot \mathbf{W}^{sk}(k)^\top \tag{7}$$

$$\hat{s} = \gamma \cdot \sigma(score_s) \tag{8}$$

$$\hat{a} = a \odot \hat{s} \tag{9}$$

$$o_i = \sum_{j=1}^{V} \hat{a}_{ij} \mathbf{W}^{v}_{j\cdot}(v) \tag{10}$$

ただし、$\mathbf{W}^{sq} \in \mathbb{R}^{Q\times d}, \mathbf{W}^{sk} \in \mathbb{R}^{K\times d}, \hat{s} \in \mathbb{R}^{Q\times K}$ であり、式 8 がスムージングの強さを表すパラメータを算出する部分である。$\sigma(\cdot)$ は入力の要素ごとにシグモイド関数を適用する関数、$\gamma \in \mathbb{R}$ はスムージングの値域を調整するハイパーパラメータである。

式 9 では、$a \in \mathbb{R}^{Q\times K}, \hat{s} \in \mathbb{R}^{Q\times K}$、$\odot$ は要素積である。このようにすることで、スムージングの強さ $\hat{s}$ を訓練データからモデルで学習させることができる。

4　実験

本実験では、Vaswani ら [3] の Transformer をベースラインにして提案手法の有効性を検証する。

4.1　実験データ

本実験では、WAT の ASPEC 英日翻訳タスクの訓練データセットの一部である train1.txt（1,000,000 文対）を訓練データとして用いた。英日両方において文長が 50 以下のデータだけを用いて学習した。用いた訓練データ数は 907,356 文対であった。検証データとテストデータは ASPEC 英日翻訳タスクの検証データセットとテストデータセットを使った。検証データとテストデータは、それぞれ 1,790 文対と 1,812 文対であった。英語文の形態素解析は WAT の手順にしたがい、日本語文の形態素解析は KyTea[1] を使った。

訓練データ 1,000,000 文対において出現頻度が 3 以下の単語は語彙辞書には登録せず、辞書に登録されていない単語は特殊トークン<UNK>に置き換えた。

4.2　モデル設定

提案モデルのパラメータはベースラインモデル[2] にあわせた。エンコーダーとデコーダーのレイヤー数は 6、ドロップアウトの確率は 0.1、ウォーミングアップステップ数は 4,000、バッチサイズは 200、エポック数は 20 とした。オプティマイザーは Adam を用い、学習パラメーターは $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 1e-9$、学習率減衰は 0.0001 に設定した。アテンションスムージングでは $s = 0.9$、ゲートアテンションでは $\gamma = 2$ とした。

4.3　実験結果

実験結果を表 1 に示す。表 1 より、ベースラインモデルと比較して、アテンションスムージングでは +0.74 BLEU(%)、ゲートスムージングでは +0.99 BLEU(%) の性能向上が確認できた。

ただし、ゲートスムージングはレイヤーごとに新しいパラメータが追加されており、ベースラインモデルよりもパラメータ数が増えている。その影響を排除して、ゲートスムージングの効果を確認するためダミーパラメータモデルを用意した。ダミーパラメータモデルは下記の計算式に従ってアテンションの計算を行うモデルであり、ベースラインモデルのパラメータ数を単純に増やしたモデルとなっている。

$$score_s = \mathbf{W}^{sq}(q) \cdot \mathbf{W}^{sk}(k)^\top \tag{11}$$

$$\hat{s} = softmax(score_s) \tag{12}$$

$$\hat{a}_{ij} = (a_{ij} + \hat{s}_{ij})/2 \tag{13}$$

$$o_i = \sum_{j=1}^{V} \hat{a}_{ij} \mathbf{W}^{v}_{j\cdot}(v) \tag{14}$$

ここで、式 12 と式 13 がゲートスムージングモデルとの差分となる。ゲートスムージングでの式 8 が式 12 に対応している。式 13 では a と $\hat{s}$ を要素ごとに

1）http://www.phontron.com/kytea/index-ja.html
2）https://github.com/jadore801120/attention-is-all-you-need-pytorch/tree/master/transformer

This work is licensed by the author(s) under CC BY 4.0
(https://creativecommons.org/licenses/by/4.0/).

表 2　a と s の値

ソフトマックス (a_j)							
0.06	0.11	0.33	0.03	0.22	0.09	0.07	0.07
0.02							
シグモイド ($\hat{s}_j$)							
0.95	0.96	0.93	0.96	0.95	0.94	1.00	0.98
0.98							

足して 2 で割る。表 1 より、ゲートスムージングはダミーパラメータモデルよりも性能が高いことがわかる．この結果から、パラメータ数にかかわらず、式 8、9 の効果（ゲートスムージングの効果）が確認できた。

4.4　ゲートスムージングの効果

ゲートスムージングにおいて学習されたスムージングに関するゲート値の実例を表 2 に示す。表 2 は、“By deciding strategy , industry-government-university should unitarily promote .” という入力文に対して、ピリオド “.” との関連性を計算した際に、エンコーダー側の最終レイヤーで算出された a_j と $\hat{s}_j$ である。表 2 より、関連性の強い a_3（strategy に対する関連）や a_5(industry-government-university に対する関連) に対し、低めの $\hat{s}_3 = 0.93$, $\hat{s}_5 = 0.95$ が与えられ、関連性の弱い a_7, a_8, a_9 に対し、$\hat{s}_7 = 1.00$, $\hat{s}_8 = 0.98$, $\hat{s}_9 = 0.98$ という高い値が与えられた。このことから全体的な分布を滑らかにする方向に学習していることが確認できた。

5　おわりに

本研究では、アテンションでの関連性計算において数単語にきわめて高い確率が与えられるという現象を緩和するため、アテンションスムージングという手法を提案した。アテンションスムージングでは人手でスムージングのパラメータを決める必要があるが、このスムージングのパラメータをモデルに学習させる手法としてゲートスムージングを提案した。WAT の ASPEC 英日翻訳タスクで提案手法の有効性を確認した。

今後の課題として、スムージングの関数を変えるなど、より良い形で関連性を調整できる式を作りたい。

謝辞

本研究成果は，国立研究開発法人情報通信研究機構の委託研究により得られたものである．また，本研究の一部は JSPS 科研費 20K19864 の助成を受けたものである．ここに謝意を表する．

参考文献

[1] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *Proceedings of the 27th International Conference on Neural Information Processing Systems*, 2014.

[2] Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *ICLR 2015*, 2016.

[3] Ashish Vaswani, Noam Shazeer, and Niki Parmar et.al. Attention is all you need. *31st Conference on Neural Information Processing Systems*, 2017.

[4] Yingzhu Zhao, Chongjia Ni, and Cheung-Chi Leung et.al. Cross attention with monotonic alignment for speech transformer. *INTERSPEECH 2020*, 2020.

[5] Hongyi Cui, Shohei Iida, Po-Hsuan Hung1, Takehito Utsuro1, and Masaaki Nagata. Mixed multi-head self-attention for neural machine translation. *Proceedings of the 3rd Workshop on Neural Generation and Translation*.

[6] Yekun Chai, Shuo Jin, and Xinwen Hou. Highway transformer: Self-gating enhanced self-attentive networks. *ACL 2020*, 2020.

[7] Shuhao GU and Yang FENG. Improving multi-head attention with capsule networks. *NLPCC 2019*, 2019.

[8] Shohei Iida, Ryuichiro Kimura, and Hongyi Cui et.al. Attention over heads: A multi-hop attention for neural machine translation. *ACL 2019*, 2019.

[9] Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. Attention is not only a weight: Analyzing transformers with vector norms. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, p. 7057–7075, 2020.

[10] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. pp. 2818–2826, 2016.

This work is licensed by the author(s) under CC BY 4.0
(https://creativecommons.org/licenses/by/4.0/).