BERTを超えたXLNetの紹介

25. June 2019

Source: Deep Learning on Medium

akihiro_f
Jun 24

この記事では、「BERTを超えた！」と話題のXLNetとその基幹となっている技術について説明します。

概要

https://arxiv.org/abs/1906.08237

XLNetは2019/6/19に、”XLNet: Generalized Autoregressive Pretraining for Language Understanding”と題してArxivに投稿された論文です。一言(?)でいうと

> *Transformer-XL*を単語の順序を入れ替えた（元の順序情報は保持）もので学習させることで、自己回帰モデルで双方向の意味依存関係を取得できるようにしたと主張。*20*を超えるタスクで*BERT*超え。

といったところでしょうか。この記事では、背景となる関連技術も含めてXLNetの着想と技術について順々に説明していきます。

自然言語処理のタスク学習の流れと事前学習の種類

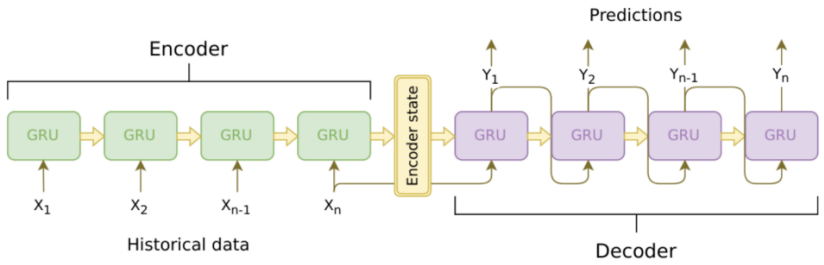
深層学習を用いた自然言語処理は翻訳、QAタスク、文書分類など多岐にわたります。

深層学習でそれらのタスクを解く際は、タスクに跨って有用な表現を教師なしで取得する「事前学習」と、事前学習の結果をもとにタスク用に再学習させる「微調整(fine-tuing)」という２段階にわけるという手法が、深層学習が発展して以降メジャーなアプローチ方法となっています。

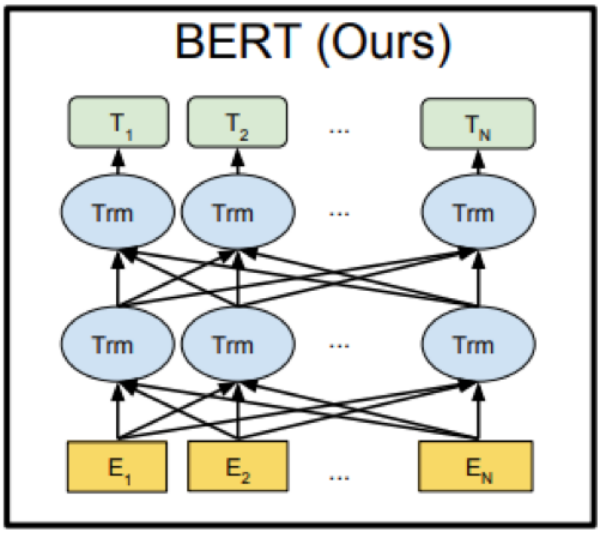
事前学習はBERTのような自己符号化(autoencoding)をもとにした手法と、LSTMを用いたSeq2Seq、GPTのような自己回帰を用いた言語モデルの２つが頻繁に使われるようです。

自己符号化をもとにしたBERTは、順方向・逆方向の情報をうまく扱えますが、予測対象の単語同士の依存関係を学習しにくいという特徴があります。

一方、LSTM等のRNN系ネットワークを用いたSeq2Seqのような自己回帰言語モデルは順々に単語を読み込ませるため、予測対象の単語同士の依存関係を学習できますが、順方向・逆方向の情報を同時に扱えないという問題がありました。



GRUを用いた自己回帰言語モデル。単語を順々に読込み、順々に予測する (画像の引用元:https://jeddy92.github.io/JEddy92.github.io/ts_seq2seq_intro/)



自己符号化をもとにした事前学習手法であるBERT。 (画像の引用元(https://arxiv.org/pdf/1810.04805.pdf))

今回紹介するXLNetは、予測対象の単語同士の依存関係を学習できる自己回帰言語モデルでありながら、自己回帰言語モデルの弱点でありBERTの良いところである「順方向・逆方向の情報を同時に扱える」性質を持っています。つまり、両者のいいとこ取りをした手法になっていると著者たちは主張しています。

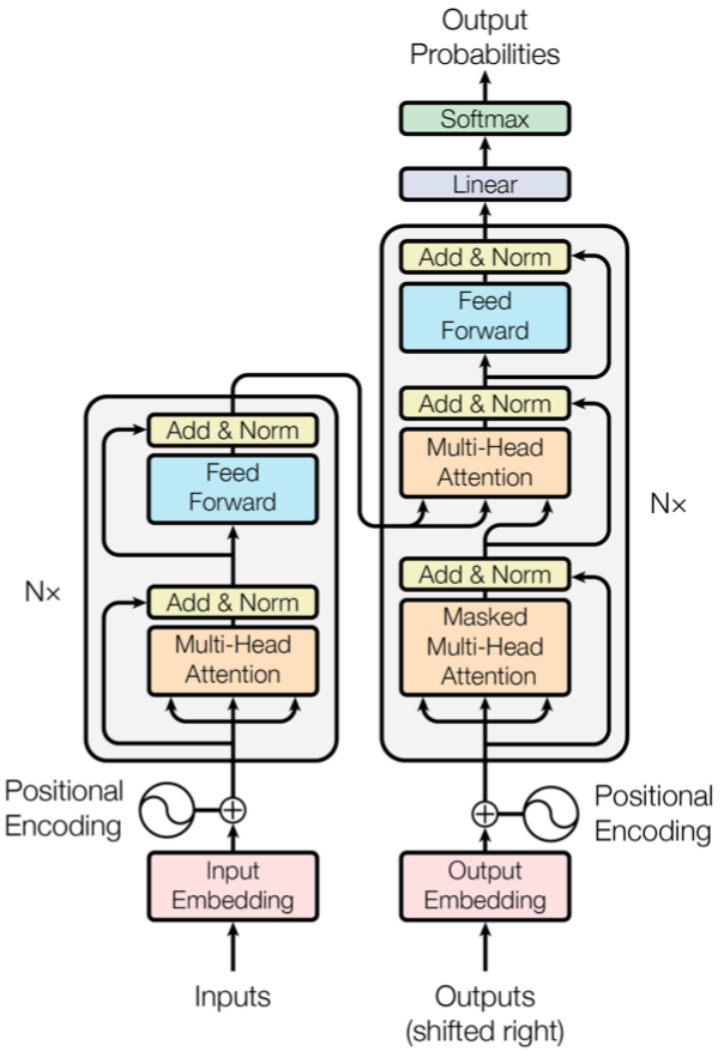
XLNetと関連研究

Transformer

https://arxiv.org/pdf/1706.03762.pdf

“Attention Is All You Need”というLSTM・CNNを愛用していた人たちに対する挑発的なタイトルでも話題になった論文で提案されたモデルです。

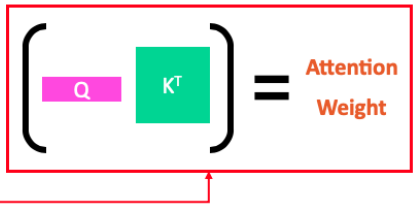
CNNでもLSTMでもないdot product Attentionという機構を提案し、それを積み重ねたモデル(Transformr)で既存手法を大きく上回る成果を上げました。この機構はXLNetだけでなく最近話題になったGPT、BERTにも使われています。



Transformerの全体図。Multi-Head Attentionとposition-wise Feed Forward Networksを組み合わせたブロックをEncoder, Decoderそれぞれで 6 つ積層させたモデル。

Transformerで使われる(dot-product) Attentionでは、Query, Key, Valueの 3 つの変数を使います。端的にいえば、Query単語とKey単語を関連性(Attention Weight)を計算し、それぞれのKeyに紐づくValueをかけるという仕組みです。

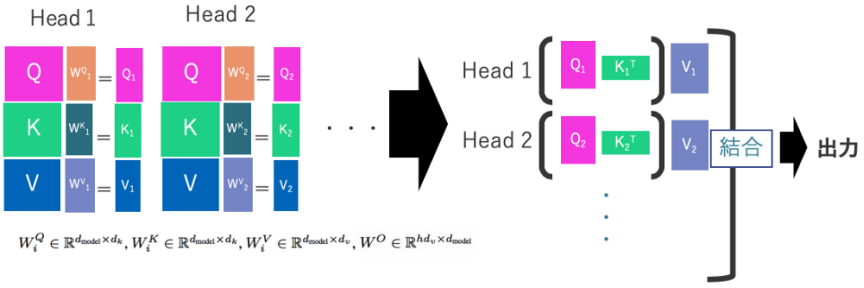
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



dot product Attention

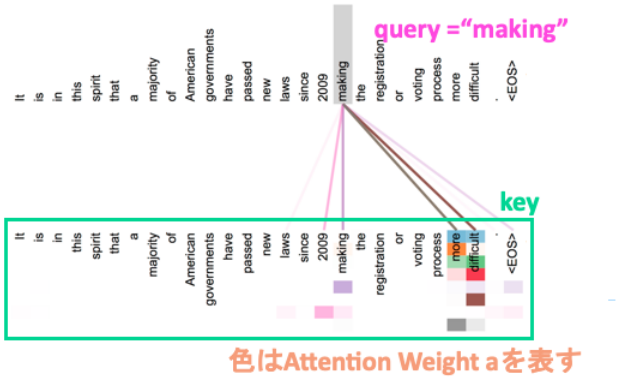
それをAttention Headを複数用いた（全結合でいうと"隠れ層の数"を増やした）Multi-Head Attentionは以下のように定義されます。上図の(Sigle Head) AttentionはQ,Kをそのまま使って

$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head}_1,...,\mathrm{head}_\mathrm{h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$



このdot product Attentionで使うQ,K,Vを同じデータからもってくるとSelf-Attentionと呼ばれます。TransformerのEncoder部分や、Decoder部分の最初のAttentionがそれにあたります。Decoder部分の上の方はQueryをEncoderから、K,VをDecoderから持ってきているのでSelf-Attentionではありません。

実際に適用したイメージを図で描くと以下の図にようになります。この図は、makingという単語をqueryとしてそれぞれのkey単語に対するAttention Weightを算出したものを可視化したものになっています。TransformerではMulti-Head Attentionを用いて後ろの層に伝播させており、それぞれのヘッドは異なった依存関係を学習しています。下図のkeyの単語に複数の着色がされていうますが、それぞれのヘッドのAttention Weightを表したものになっています。



Transformer を文に適用した際の模式図

Transformerの詳細な内容については、このブログが分かりやすいのでオススメです。

Transformer-XL

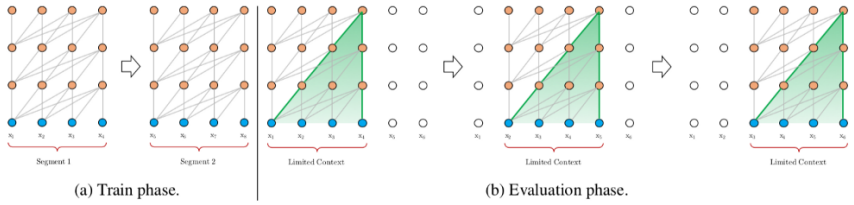
https://arxiv.org/pdf/1901.02860.pdf

しかし、Transformerはdot-product Attentionの性質上、固定長しか扱えません。理想的には長大な固定長を持つモデルを構築すれば対応できますが、Transformerは重量級モデルなので、あまり現実的な選択肢ではないでしょう。この問題を解決しようとしたのがTransformer-XLです。対象文書の前部分(Transformerでは参照できない部分)でgradientをとらないようにすることによって、計算グラフを固定したまま文書全体を参照しながら計算値を出せるようにしています。

その戦略を可能にするために、”Segment-Level Recurrence with State Reuse”と”Relative Positional Encodings”という 2 つの手法を提案しています。

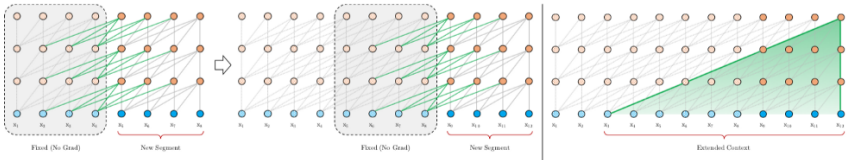
Segment-Level Recurrence with State Reuse

雑にまとめると、「過去のシーケンスの情報の勾配をとらないことで計算グラフは固定長のままにするが、値はメモリに残しておいて参照する」手法です。この手法をとることで、通常のTransformerが固定長以上のシーケンス情報を扱えないのに対し、固定長以上のシーケンス情報を扱えるようになります。



(a) Train phase. (b) Evaluation phase.

通常のTransformerの学習・推論。固定長(=4)の情報しか扱えない。



(a) Training phase. (b) Evaluation phase.

Transformer-XLの学習と推論。過去の情報を固定して使うことで、固定長以上の情報を扱える。

Relative Positional Encodings

雑にまとめると、位置に対する学習パラメータを導入することで、位置関係の影響を”学習”できるようになったことが改善点のようです。

通常のTransformerとTransformer-XLのMulti-Head Attentionを定式化すると以下のように書けます。違いとしては、Key側の位置エンコーディング項U_jが相対値なっている点、、Query側の位置エンコーディング項U_iが学習パラメータu, vに置き換わっている点、c項・d項のW_kがW_(k,E),W_(k,R)に置き換わっている点、です。

$$\mathbf{A}^{\text{abs}}_{i,j} = \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{E}_{x_j}}_{(a)} + \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{U}_j}_{(b)} + \underbrace{\mathbf{U}_i^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{E}_{x_j}}_{(c)} + \underbrace{\mathbf{U}_i^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{U}_j}_{(d)}.$$

通常のTransformerのMulti-Head Attention

$$\mathbf{A}^{\text{rel}}_{i,j} = \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_{k,E} \mathbf{E}_{x_j}}_{(a)} + \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_{k,R} \mathbf{R}_{i-j}}_{(b)} + \underbrace{u^\top \mathbf{W}_{k,E} \mathbf{E}_{x_j}}_{(c)} + \underbrace{v^\top \mathbf{W}_{k,R} \mathbf{R}_{i-j}}_{(d)}.$$

Transformer-XLのMulti-Head Attention

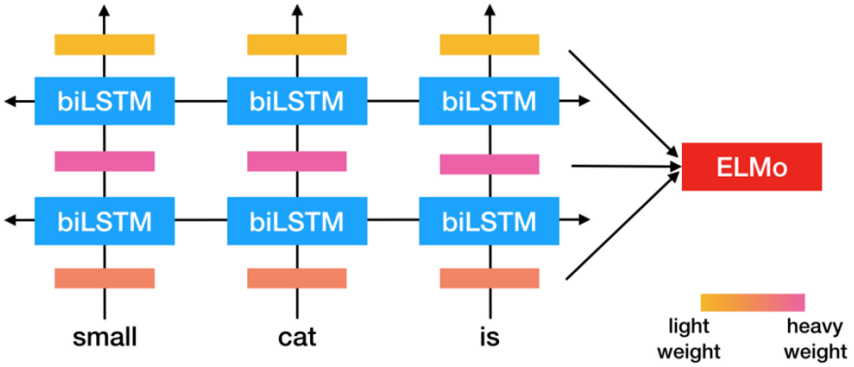
c項で文字情報項の大きさを、d項で相対位置情報の大きさを学習できる仕組みになっています。

ELMo

https://arxiv.org/pdf/1802.05365.pdf

双方向の情報を情報を使って、文脈を考慮した分散表現を得る手法を提案した手法です。具体的にはBi-directional LSTMの中間層を重み付けして結合したものを分散表現として用います。

順方向・逆方向両方の情報を用いているので、文脈を考慮した分散表現が取得できるとELMoの著者たちは主張してます。



図はhttps://kamujun.hatenablog.com/entry/2018/07/06/173931から引用

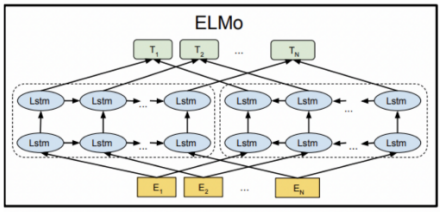
BERT

https://arxiv.org/pdf/1810.04805.pdf

言わずと知れた2018年の有名モデルです。ELMoでは双方向の情報を使っているが、実際は「予測先が見えてはいけない」という制約がります。

具体的には、”I play tennis every day”という文の”tennis”部分を予測する場合は、順方向でtennisの次に予測対象になる”day”が逆方向では見えてしまいます。この状態では上手く予測モデルを作れないため、ELMoでは順方向と逆方向を別々に学習させた後に活用するという手法をとっていました。

順方向：$\log p\,(tennis \mid I\ play)$
逆方向：$\log p\,(tennis \mid day\ every)$

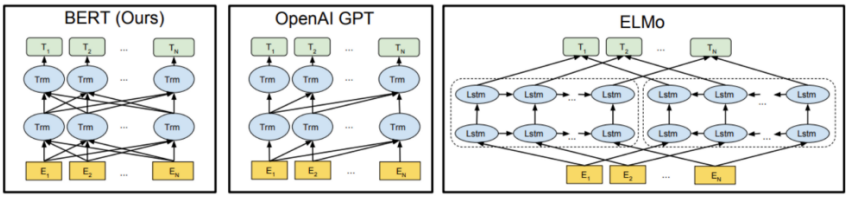


(左)順方向・逆方向それぞれの方向から"tennis"を予測した際の条件付き確率 (右)ELMoの模式図

BERTの著者達は、予測したい単語を[mask]にすることで順方向・逆方向の両方の情報を同時に使う手法を提案しました。（実際は[mask]だけで代替するわけではないですが、本筋ではないため詳細は原論文を参照してください）

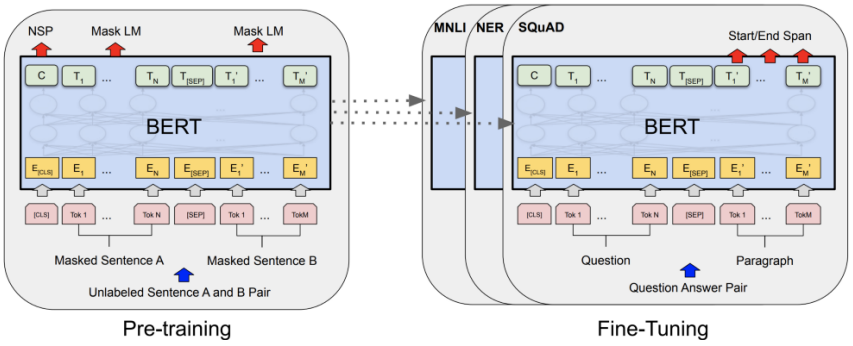
入力文は下記のようになり、順方向から読み込んでも逆方向から読み込んでも、予測対象の情報が漏洩しないという特色があります。

さらに先行研究として存在していたOpenAIのGPTモデル のように双方向LSTMの代わりに双方向Transformer Encoderを使ってうことで、モデルの表現力を大幅に向上させています。



BERT, GPT, ELMoの比較

BERTは様々なタスクにおいてstate-of-the-artを更新しただけではありません。BERTには「事前学習した後にモデルはそのままで多様なタスクにそのまま応用できる」という優れた点があります。本筋ではないので詳細は省きますが、最終層の出力の仕方を変えるだけで、色々なタスクにそのまま応用できる工夫がなされています。この点もBERTが話題になった理由の１つでしょう。



XLNet

さて、やっと本題のXLNetに入ります。

多くの素晴らしい功績を残したBERTですが、XLNetの著者達は２つの問題点を指摘しています。

1. [mask]という特殊な文字を使用しているので、それらが出現しないfine-tuning時に問題がでてくる
2. 後述するようにBERTのような自己符号化モデルでは、予測対象の単語が複数あった場合、自己回帰モデルのように予測対象の単語間の依存関係をとることができない

著者たちは、[mask]を必要としない自己回帰言語モデルを使いながらも双方向の情報を同時に扱える手法と、それにTransformer-XLの技術を加えたXLNetというモデルを提案しています。

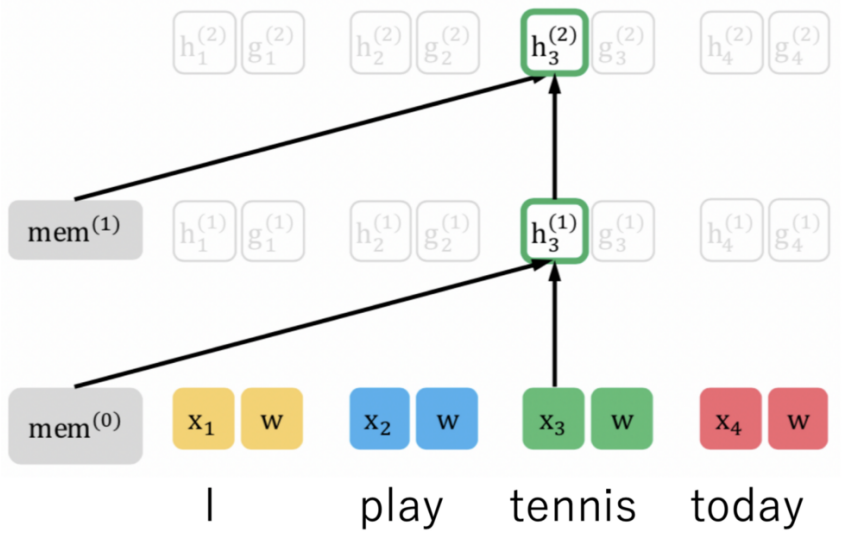
単語の予測順序の入れ替え

自己回帰言語モデルの弱点である「双方向の情報を同時に扱えない」という点を克服するために、「単語の予測順序を入れ替える」手法をXLNetでは使っています。しかし、単語を入れ替えた文書をそのまま入れるのではなく、Transformer-XLで提案されたRelative Positional Encodingsを使って元の文書における位置情報を残しておくことがポイントです。

例として”I play tennis today”という文を予測する場合を考えます。

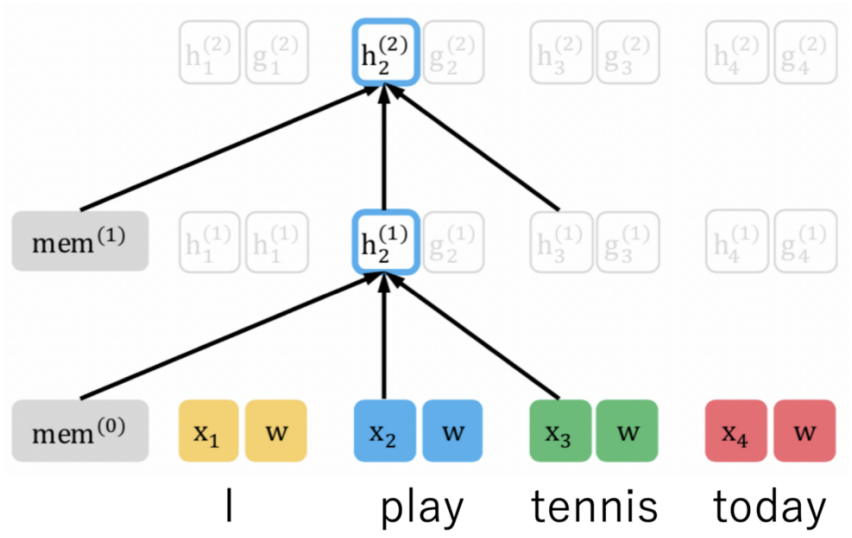
XLNetでは、この文の単語順を入れ替えた[tennis, play, today, I]という順序で予測を実施しますが、元の文における位置情報[3,2,4,1]をTransformer-XLで提案されたRelative positional encodingを使って付与します。

まず、入れ替えた順序でいうと2番目に位置する”play”を予測する場合を考えてみます。入れ替えた順に予測するので、入れ替えた後に１番目に位置（元の文書で３番目に位置）する”tennis”の情報とメモリに保存した情報を扱います。メモリに保存されているのはTransformer-XLで**”Segment-Level Recurrence with State Reuse”**(勾配はとらないが情報は使う)の部分です。



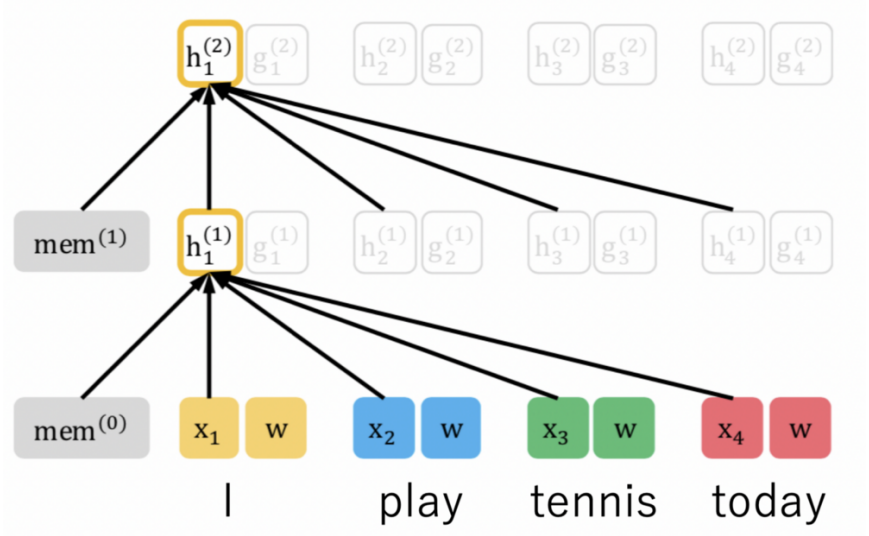
“I play tennis today”を[tennis, play, today, I]に並び変え、[tennis]からplayを予測する

次に入れ替えた後の3番目に位置する”today”を予測するために、入れ替えた後に１,2番目に位置（元の文書で３,2番目に位置）する”tennis”,”play”とメモリの情報を扱います。



“I play tennis today”を[tennis, play, today, I]に並び変え、[tennis, play]からtodayを予測する

次に入れ替えた後の4番目に位置する”I”を予測するために、入れ替えた後に1,2,3番目に位置（元の文書で3,2,4番目に位置）する”tennis”,”play”,”today”とメモリの情報を扱います。



“I play tennis today”を[tennis, play, today, I]に並び変え、[tennis, play, today]からIを予測する

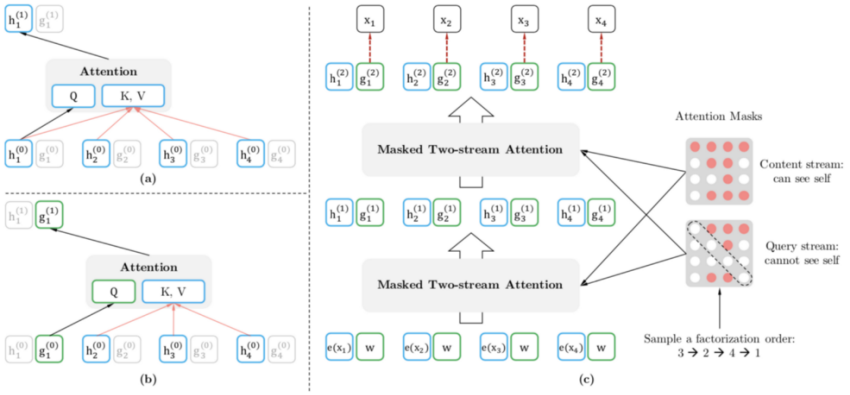
この文はここで終わりですが、文に続きがある場合はTransformer-XLのように、最初の部分を勾配を取らないパラメータとして固定（メモリに入れる）し、ずらして文の続きを学習させます。もちろん文の続きとは、元の文の順番の続きではなく「並び替えられた文の続き」です。

Two Stream Self-Attention

基本的にはTransformer-XLと同様にTransformer Encoder(Self-Attention)をモデルで用います。しかし入れ替えにより単語順序がランダムになっているため、このままでモデルを構築すると、次の単語の出現確率はどの単語でも同じになってしまい、上手く学習できません（入れ替わりがあるため、”次”という位置に意味がない）。

そのため著者達は通常のSelf-Attention(Content Stream Attention)の他にQuery stream Attentionというものを導入しています。

下図がそれを模式的に表したもので、潜在変数hとgという２つを用いてContent Stream Attention、Query stream Attentionを構築しています。



(a)Content Stream Attention, (b)Query stream Attention (c) overview

Content Stream AttentionとQuery Stream Attentionの違いは、前者は参照位置の単語情報を直

例として、”I play tennis today” の並びを入れ替えた[tennis, play, today, I]という文を考えます。(この場合の位置zのシーケンスは[3,2,4,1]となり、その情報をTransformer-XLで提案されたrelative positional encodingを使って付与されています）

(a)のようにContents Stream Attentionの場合は、読み込み済みの[tennis, play, today]（位置:[3,2,4]）に対応する潜在変数h_3, h_2, h_4(単語情報に相当)だけでなく、参照位置の単語である”I”（位置:[1])に対応する潜在変数h_1(単語情報に相当)を参照することができます。参照位置の単語である”I”を含む全ての情報を使ってKey, Valueを構成し、Queryは参照位置の単語である”I”に対応する潜在変数h_1を使ってAttentionをかけています。

一方、Query Stream Attentionの場合は参照位置以外の[tennis, play, today]（位置:[3,2,4]）に対応する潜在変数h_3, h_2, h_4(単語情報に相当)のみでKey, Valueを構成し、Queryを参照位置（位置:[1]）に対応する潜在変数g_1（位置のみ、単語情報なし）からとってきてAttentionをかけています。

BERTとの比較

まず、順方向・逆方向両方の情報を扱えるBERTとXLNetを比較を、具体例として”New York is a city”という文を推論する場合で考えてみます。

BERTでは、予測対象の単語にマスクをかける([mask]という特殊文字で置き換える）ので、入力文は以下のようなります。

[mask] [mask] is a city

一方XLNetでは、文の入れ替えを行うため以下のようになります。（ここでは例として”is a city New York”という入れ替え方になったとします）

New予測時の入力： *is a city*

York予測時の入力：*is a city New*

よって、BERT, XLNetの目的関数は以下のように書くことができます。ここで注目してほしいのは、XLNetの目的関数の第２項で予測対象の単語である”New”と”York”の依存関係が現れているところです。著者達は予測対象の単語同士の依存関係が学習できる点を、BERTと比較した際のXLNetの利点として挙げています。

$$\mathcal{J}_{\text{BERT}} = \log p(\text{New} \mid \text{is a city}) + \log p(\text{York} \mid \text{is a city})$$
$$\mathcal{J}_{\text{XLNet}} = \log p(\text{New} \mid \text{is a city}) + \log p(\text{York} \mid \text{New}, \text{is a city})$$

BERT, XLNetの目的関数

自己回帰言語モデルとの比較

次に自己回帰言語モデルとXLNetを比較しています。

通常の自己回帰モデルは順方向、もしくは逆方向に文書を読み込ませるため、片方向の依存関係しか学習することができません。

例えば、”New York is a city”を順方向に読み込ませる場合だと、*p(York | New)* は学習できますが、*p(New | York)*は学習できません。ELMoのように順方向・逆方向両方の情報を扱う手法もありますが、前述のように事実上別々に学習させているため不十分そうです。

一方、XLNetでは学習時に単語の予測順序の入れ替えを行うことによって、順方向・逆方向両方の依存関係を学習することができます。

結果

まず、RACEというデータセットでは、BERT-Large(モデルが大きな高性能Ver.のBERT)を超える結果となっています。

RACE	Accuracy	Middle	High
GPT [25]	59.0	62.9	57.4
BERT [22]	72.0	76.6	70.1
BERT+OCN* [28]	73.5	78.4	71.5
BERT+DCMN* [39]	74.1	79.5	71.8
XLNet	**81.75**	**85.45**	**80.21**

Table 1: Comparison with state-of-the-art results on the test set of RACE, a reading comprehension task. ∗ indicates using ensembles. "Middle" and "High" in RACE are two subsets representing middle and high school difficulty levels. All BERT and XLNet results are obtained with a 24-layer architecture with similar model sizes (aka BERT-Large). Our single model outperforms the best ensemble by 7.6 points in accuracy.

SQuADではBERTのアンサンブルを単一モデルで超えています。

SQuAD1.1	EM	F1	SQuAD2.0	EM	F1
Dev set results without data augmentation					
BERT [10]	84.1	90.9	BERT† [10]	78.98	81.77
XLNet	**88.95**	**94.52**	XLNet	**86.12**	**88.79**
Test set results on leaderboard, with data augmentation (as of June 19, 2019)					
Human [27]	82.30	91.22	BERT+N-Gram+Self-Training [10]	85.15	87.72
ATB	86.94	92.64	SG-Net	85.23	87.93
BERT* [10]	87.43	93.16	BERT+DAE+AoA	85.88	88.62
XLNet	**89.90**	**95.08**	XLNet	**86.35**	**89.13**

Table 2: A single model XLNet outperforms human and the best ensemble by 7.6 EM and 2.5 EM on SQuAD1.1. ∗ means ensembles, † marks our runs with the official code.

他にも色々なデータセットでstate-of-the-art更新しており、結果20タスクでBERTを超える結果を示しています。

Model	IMDB	Yelp-2	Yelp-5	DBpedia	AG	Amazon-2	Amazon-5
CNN [14]	-	2.90	32.39	0.84	6.57	3.79	36.24
DPCNN [14]	-	2.64	30.58	0.88	6.87	3.32	34.81
Mixed VAT [30, 20]	4.32	-	-	0.70	4.95	-	-
ULMFiT [13]	4.6	2.16	29.98	0.80	5.01	-	-
BERT [35]	4.51	1.89	29.32	0.64	-	2.63	34.17
XLNet	**3.79**	**1.55**	**27.80**	**0.62**	**4.49**	**2.40**	**32.26**

Table 3: Comparison with state-of-the-art error rates on the test sets of several text classification datasets. All BERT and XLNet results are obtained with a 24-layer architecture with similar model sizes (aka BERT-Large).

最後に

GithubのREADMEに記載されていますが、XLNetではTransformer-XLのように過去のシーケンスを保持するためか大量の高性能GPUが必要のようです（著者達はTPUで実験）。BERTやBigGANsもそうですが、最近のSOTAモデルは「金(大量のGPU/TPU)で殴る」ものが多い気がします。家の低性能GPUでも学習して遊べるSOTAモデルでないかなぁと切実に思う今日この頃です。

解釈が間違っている点等あれば、指摘していただけると嬉しいです！

About Coldstart.nlp and about myself.
Source: Deep Learning on Medium
Coldstart.nlpの概要と記事紹介、少しだけ自己紹介Akira TakezawaMar 14Photo by Alain Wong on UnsplashColdstart.nlpは、自然
14. March 2019
Similar post

スパコン分野の大規模国際会議SC18
Source: Deep Learning on Medium
Arai, JunyaDec 24こんにちは．NTT研究所の新井です．会社では「計算の高速化」を軸に，効率的な並列グラフアルゴリズムの研究や，深層学
25. December 2018
Similar post

Neuromation Q2第２四半期報告書
内容：● CEOからの手紙 ● はじめに ● Neuromation ラボの近況報告 ● 開発情報の更新● NTKの情報更新 CEOからの手紙　人工知能（AI）は、私たちの世界の仕組みを変えて
27. August 2018
Similar post

« NEURAL NETWORKS FOR MUSIC GENERATION

MACHINE LEARNING, AI AND BEYOND— WHAT ARE THE LIMITS OF THE HYPE? »