第三章 リカーレントニューラルネットワークと自然言語処理

本章では、自然言語処理で主に使われるRNN（リカーレントニューラルネットワーク）の基礎からより応用的で現在最新の研究に近いアテンションやBERTに関する理解を目指して説明していきます。

ここでは自然言語の話をしていきます。画像認識のようなコンピュータービジョンと自然言語が何か違うかというと、コンピュータービジョンというのはコンピューターに目のような機能を持たせるというものですが、これが扱う対象は画像を見てそれが何かというような人間の目が行うこと、つまり物理的な光が解析対象でありますが、このような解析は人間だけができるものではなくて、動物である犬とか猫なども何かものを認識できたりしますし、より小さな動物でも可能なものです。ある程度複雑ではありますが、そこまで難しすぎるタスクではないと言えると思います。それに対して、言語は人間しか認知できないようなものが対象です。たとえば、下図の上の例のように、左の鳥の画像に対してこれは鳥ですというような、実際のものとは別の人間が認知する概念としての言葉というものを人が認知するわけです。さらに図の下の例では左の赤い丸のような画像を見て、これはリンゴであると認知するわけです。

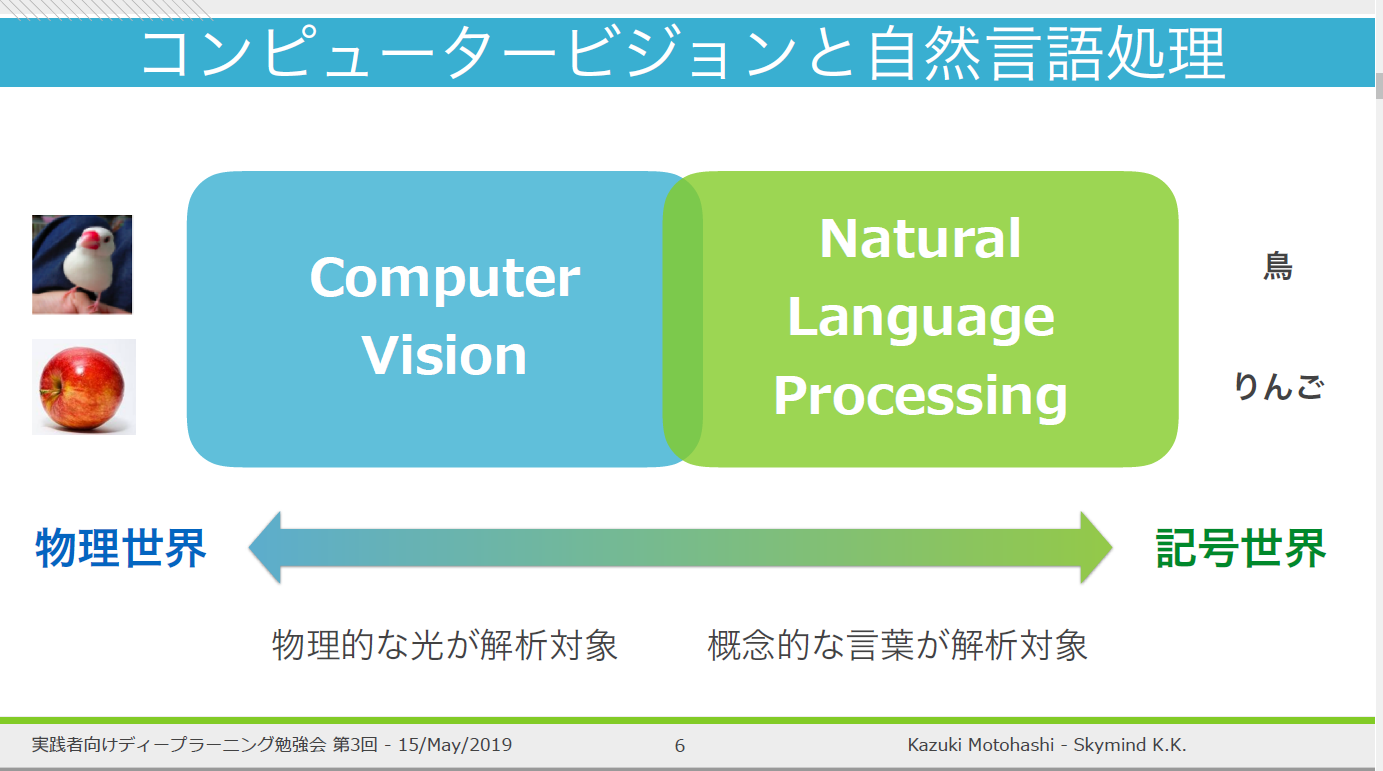
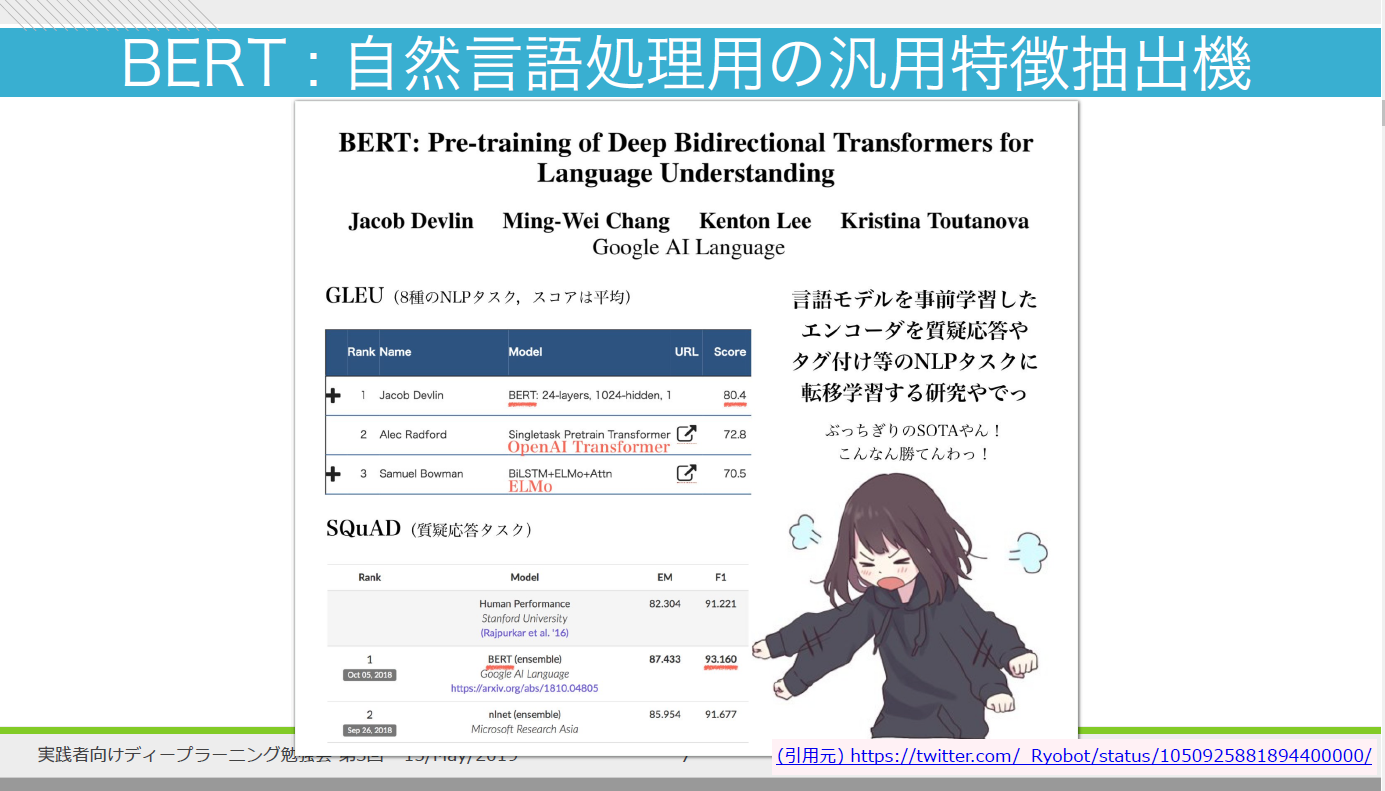


図 コンピュータービジョンと自然言語処理

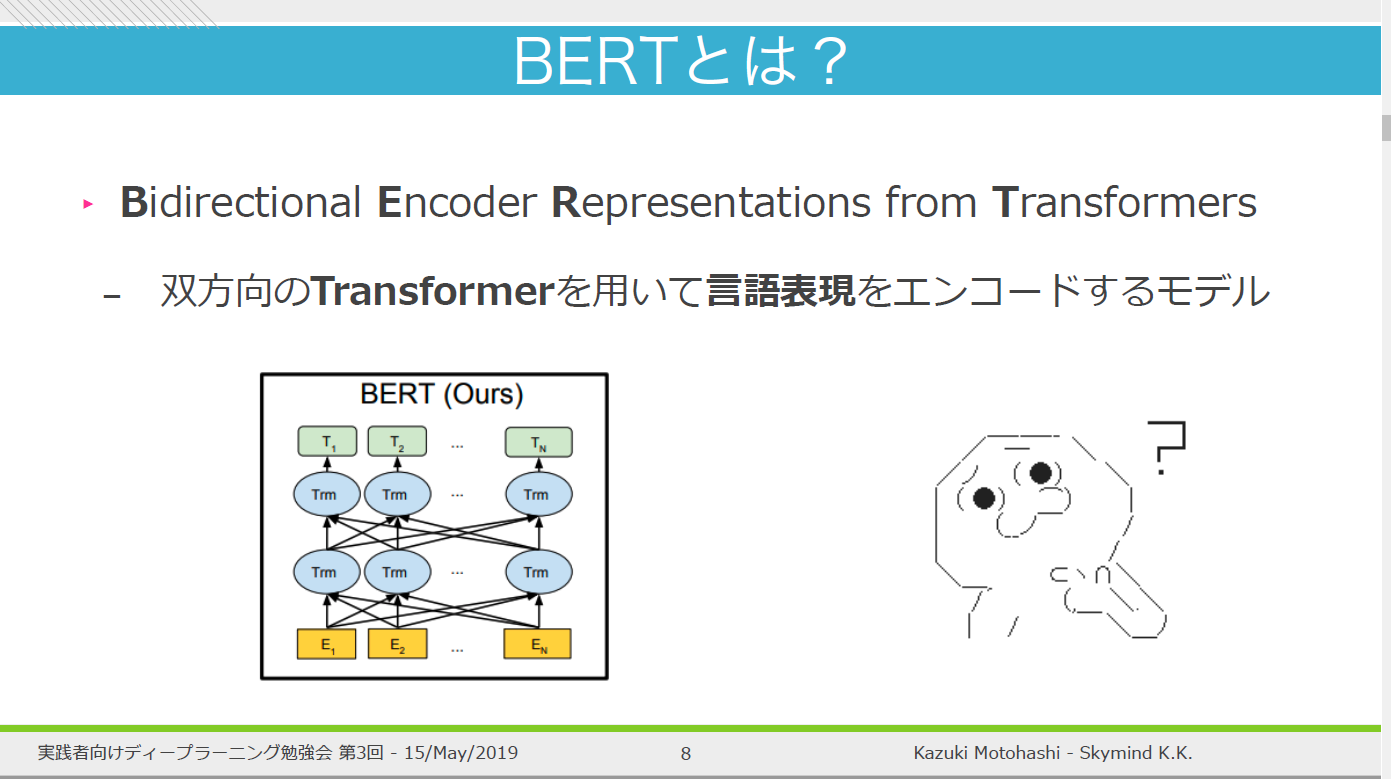
簡単にニューラルネットワークが扱うものとしてコンピュータービジョンや自然言語処理がありますが、そもそもの認知の対象が異なるというところを認識していただければと思います。そのため、コンピュータービジョンでは簡単にできていたことが、自然言語処理ではなかなか難しいという背景にはこう言った背景があります。

しかしながら、下図のように、2018年の冬にBERTという新しいモデルが出てきました。これはImageNetで行うような特徴量抽出というものを、自然言語処理でも似たようなことができるようになりましたということを言っている論文があります。そのため、この内容はツイッターなどでも大きく取り上げられて、研究以外でも会社の中でも話題になったようなモデルです。



図： BERT自然言語処理用汎用抽出器

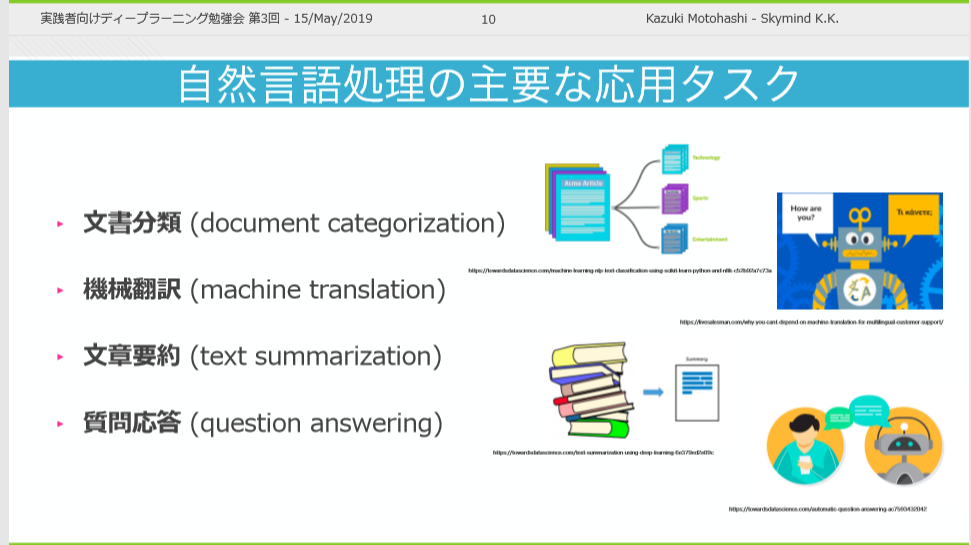
BERTが具体的には何かというと、Bidirectional Encoder Representation from Transformersの頭文字をとっていて、これを日本語で表現すると、双方向のTransformerを用いて言語表現をエンコードするモデルというものになりますが、日本語でも一見して理解できないと思います。まずこの双方向のTransformerのTransformerが何かというと、これはGoogleのニューラル機械翻訳モデルと呼ばれるもので、これから先でRNNの話をするわけですが、そのRNNを使わずにAttention機構を用いたモデルであるということですが、まだRNNやAttention機構の話をしていないため、複雑ですしわからないと思います。



図： BERTとは？

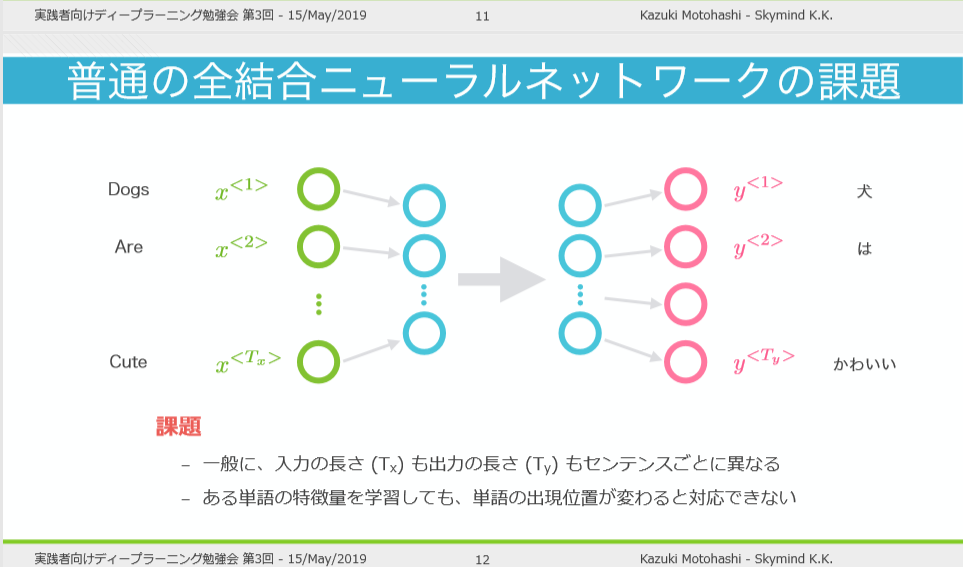
本節では、BERTの雰囲気を理解していただくことを目標として、自然言語処理とRNN, Attention機構とは何か、Transformer/BERTの概要という内容について3本立てで説明をしていきます。

最初に自然言語処理の話をします。自然言語処理とは人間が使う言葉を機械で処理をするというものです。自然言語処理の応用のタスクにどのようなものとして下図のようなものがあります。たとえばわかりやすいところで言うと、ニュース記事があって、その内容が科学技術について書かれたものなのか、スポーツのニュースなのか、エンターメント系のゴシップ系の芸能ニュースであるのかなどを分類するときに、人手の場合にはいくつかの単語を見るなどを行い簡単に分類できますが、機械で行うのは難しいところを行える文章分類があります。もっと有名なところで言うとGoogle翻訳のような機械翻訳で、英語から日本語に翻訳やまたその逆もしかりですが、そのようなタスクがあります。また、大量な文章の中でそれを短い文章にまとめるような要約を行う文章要約というタスクもあります。文章要約はつい最近論文の本文の内容からその要約を自動生成するような技術論文が公開されていました。その論文自体は非常に面白くできていて、論文にはかならず最初に記述されているアブストラクト（要約）がありますが、その論文のアブストラクト自体を論文で用いられている文章要約の技術を用いて機械的に生成しています。実際その要約自体がすごく自然な文章で書かれていますが、そのようなところまでできるようになってきています。あとはチャットボットに代表されるような質疑応答などの応用タスクがあります。



図： 自然言語処理の主要な応用タスク

このようなタスクをニューラルネットワークで実践しようという流れがありますが、普通の全結合ニューラルネットワークを使うとなかなか難しいです。なぜ難しいのかをこれから説明します。文章の翻訳タスクを解かせたいと思ったら、最初に英語の文章をニューラルネットワークに対して入力してみます。たとえば下図のように、Dogs are cute. を入力したら 犬はかわいい というような文章が出てきて欲しいというような問題設定でニューラルネットワークを訓練していきます。一般に入力部の長さや出力部の長さはセンテンスごとに異なります。今回の例では、 Dogs are cute. という3つの単語からなる入力を入れていますが、もっと短い文章であったり長い文章であったりしても良いわけです。そのため、全結合ニューラルネットワークのように固定の長さの入力があって中間層があって固定の長さの出力層があるというような全結合ニューラルネットワークだとなかなか扱いづらいです。たとえば100個の入力のユニットを用意しておいて、今回の例のように‘Dogs are cute’という3つの単語を入力してあとはブランクの情報を入れるというような仕組みでニューラルネットワークを実現することもできるかもしれませんが、ブランクが多すぎて学習が進みづらいとか、そもそも200個の単語があった場合にはどうなるかというようにどんどん対応ができなくなるためこのようなアプローチは難しいです。



図： 全結合ニューラルネットワークの課題

また上図の例でも、ある単語の特徴量である、たとえばdogは犬のようなものを表すというようなものを学習することは可能ですが、’I like a dog’ というような別の文章が入力された場合には、全結合ニューラルネットワークでは、先の例で最初dogがあったら犬だというようなものは学べますが、4番目の単語にdogがあった場合には、新しく学習し直さなければならないため、一回学んだことが次に応用できなくなってしまいます。全結合ニューラルネットワークではそのような位置の依存性が出てきてしまうために適さないということで、リカーレントニューラルネットワークという時系列の情報を持ったようなデータ、つまり順番によって内容が変わるようなデータというものに対して用いられます。リカーレントニューラルネットワークが何かというと、下図の絵は１つニューラルネットワークだと思ってください。この処理では下から上に処理が進んでいきます。まず一番下から入力があるとすると、それに対して重みをかけて、中間層の各列の状態を得て、また重みをかけて、出力を得るといった一連の処理を行っています。

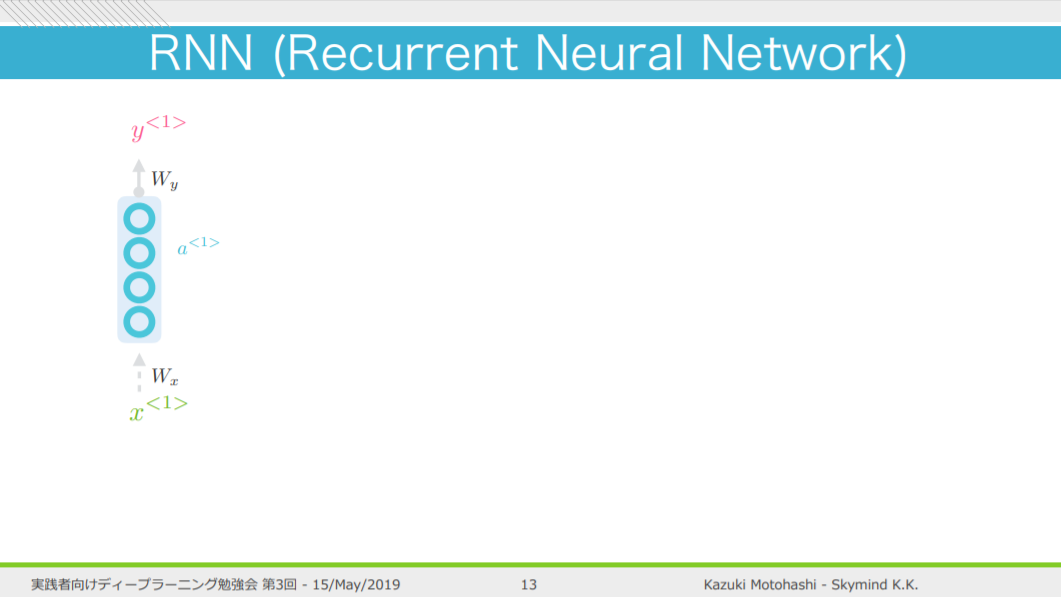


図 RNNのプロシージャ説明（１）

この処理がこれだけで終わるのではなく次の入力が入ってきた場合には、下図のように、その入力だけではなくてその１個前の状態からの中間層からの状態も入力にして行列の掛け算をして出力を得るというようなことを行います。

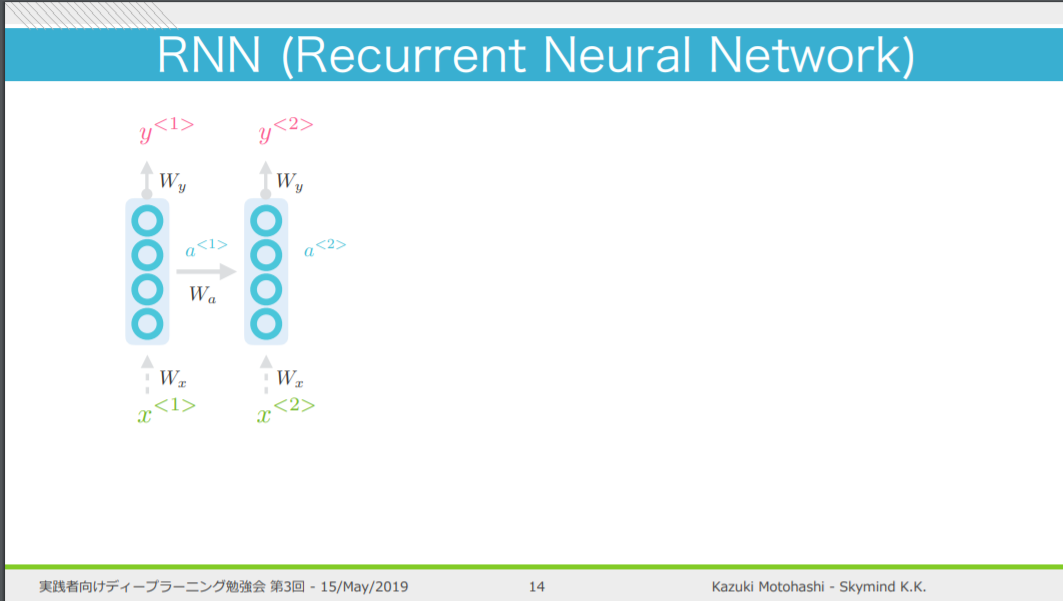


図 RNNのプロシージャ説明（２）

それをさらに下図のように繰り返していくわけです。

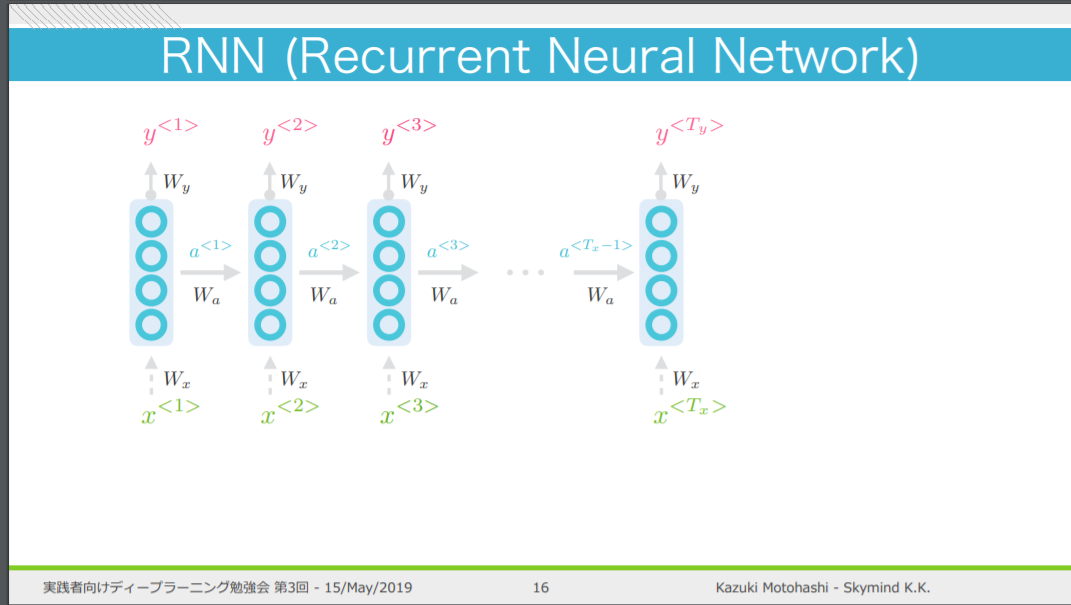
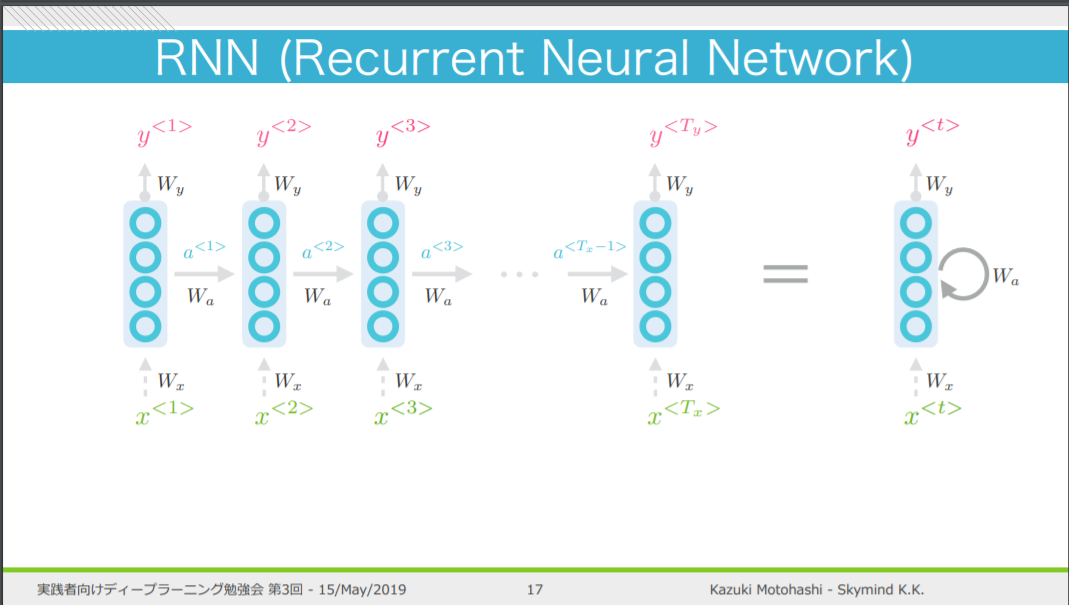


図 RNNのプロシージャ説明（３）

このようにして、今までの時系列データの情報をメモリのようなものを作って中間の状態にどんどん貯めていって、今までの入力の状態を考慮しながら次の出力を得ていくという手法がリカーレントニューラルネットワーク（RNN）で行っていることとなります。

上図のネットワークは省略して、下図の右の赤枠のように記述することがあります。リカーレントネットワークは、自分の中間の状態が次また入ってくるというようなループがあるので、そのループが再帰的であることから、再帰的（リカーレント）という呼び方がされているネットワークです。



このようなネットワークを構築すると何がうれしいかというと、すでに少しお話しましたが、下図のように、さきほどの例の’Dogs are cute’のような文章があった場合、cuteという単語を入力したときに、かわいいと出てきて欲しいのですが、リカーレントネットワークを用いるとdogs areというような前のデータの状態も考慮して最後のかわいいという答えを出すようなことが可能になります。つまり今までの入力のデータをあいまいで抽象的な状態にエンコードして出力を出すということができるようになっています。

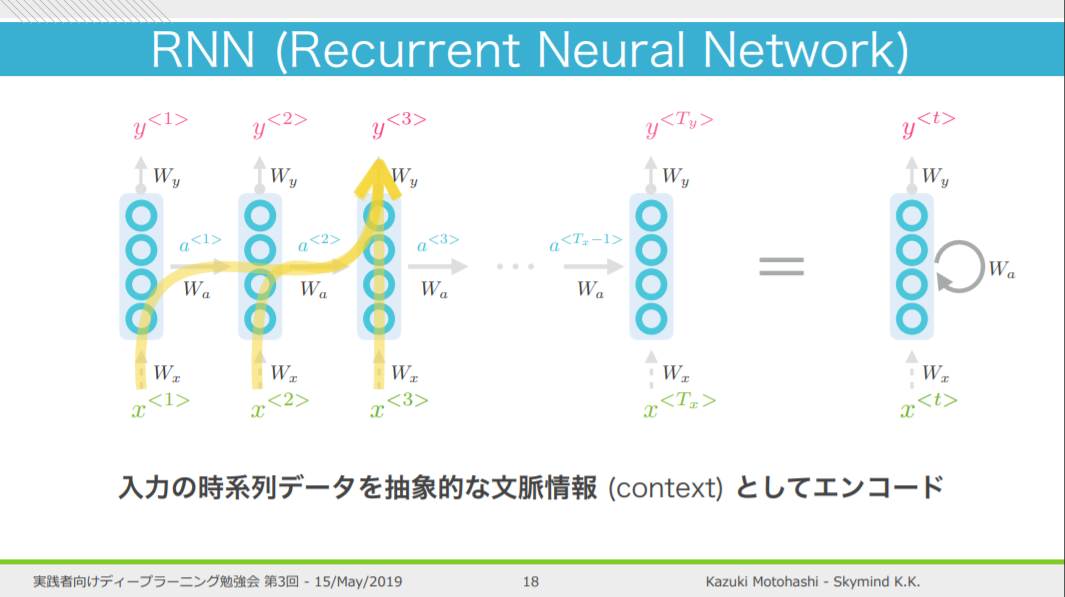


図 RNNの入力データのエンコードイメージ

RNNのユニット

それでは、RNNのユニットの詳細についてこれから見ていきます。

RNNの１つ１つのユニットはさきほどの図でいうと、４つの〇で構成される真ん中のところとなり、これを１つのユニットだと思って回路のように書き下出してみると、下図のようになります。

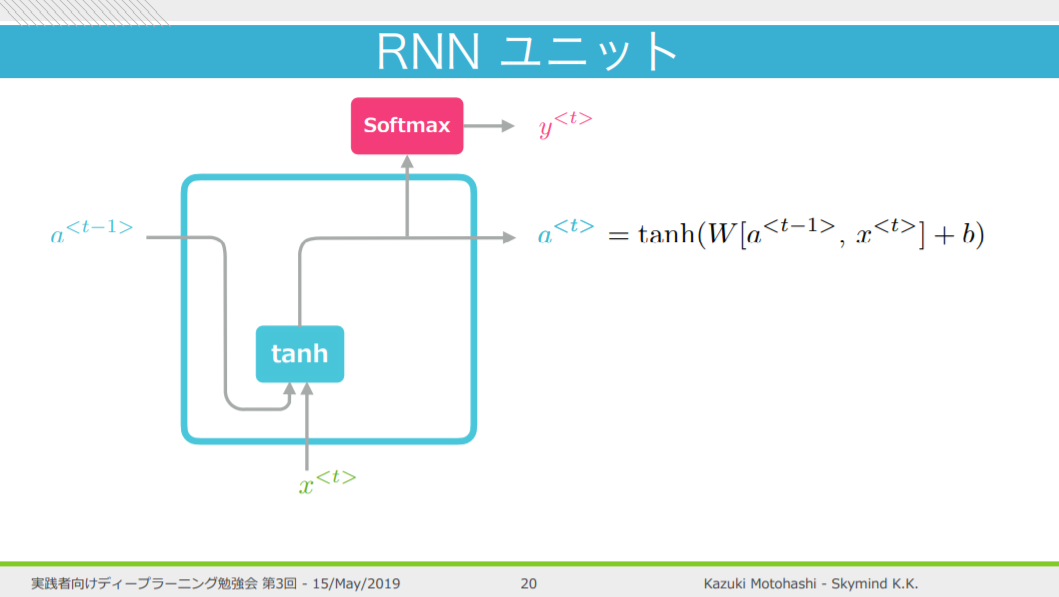


図 RNNのユニット

RNNのユニットの動作は、上図の現時点の入力のx<t>とその前の時点の中間層の状態のx<t-1>がtanhの数式で示される行列で掛け算されて活性化関数の出力を次に出すということを延々と繰り返すというものです。そこで何か出力を出したい場合に、たとえば確率的な出力を出したい場合には、上図の例のようなSoftmaxと呼ばれる活性化関数にtanh関数の計算結果を入力して、Softmaxを用いて一番確率が多いものを出力として出すようなことを行います。しかしながらこのユニットは実は簡略化されすぎています。たとえば、次の図にあるような’The cat which already ate …, [was] full’のような‘猫がいろんなものをたくさん食べた結果おなかがいっぱいです’というような文章を出力したいときに、図の[]内のbe動詞に何が入るかを予測するという課題では、今回の例では、ateという単語が前にあるので、過去の状態を表しているという条件に引きずられてwasになるというようなロジックが必要になります。ただしcatが２番目の文章のようにcatsという複数形になった場合には、さらにwereになるわけです。このような文章の場合、catとfullの間に挟まれているほかの文章は関係がなく、本文の冒頭の名詞であるcatが単数か複数化にだけに依存します。そのため、[]内の正しいbe動詞を得たいときに、中間の文章は気にしないというような操作もニューラルネットワークの機能の中に入っていて欲しいという思想が生まれ、それを実現するためのゲートが追加されたGRUと呼ばれるRNNの進化版にあたるものが提唱されています。

GRUが何かというと、今までのRNNは入力と前の中間層の状態にバイアスをプラスして活性化関数とかけたものが出力だったわけですが、それに図の右の四角のようにsigmoid関数と呼ばれる０～１の出力値を返す活性化関数が追加されています。このsigmoid関数に対しても入力と一つ前の中間層の状態を入れることで、RNNに0か１つまりTrueがFalseにあたるスイッチのオンオフ機能をつけることができるようになります。これをさきほどの通常のRNNで得られる図の左側の<t>に掛け算してあげれば、たとえば、これが１を返せば、中間層のをそのまま出力し、０であれば、次の状態に渡さないので、その前のa<t-1>の状態がそのまま出力されるというような形の情報の流れにすることができます。イメージとしては、catが入ってきて、whichとかalreadyとかate以外はすべて0を掛けて、必要となるものだけを覚えて、[]の出力の計算にその出力を使うということができるというようなものが学習できるような機構が追加されたものです。

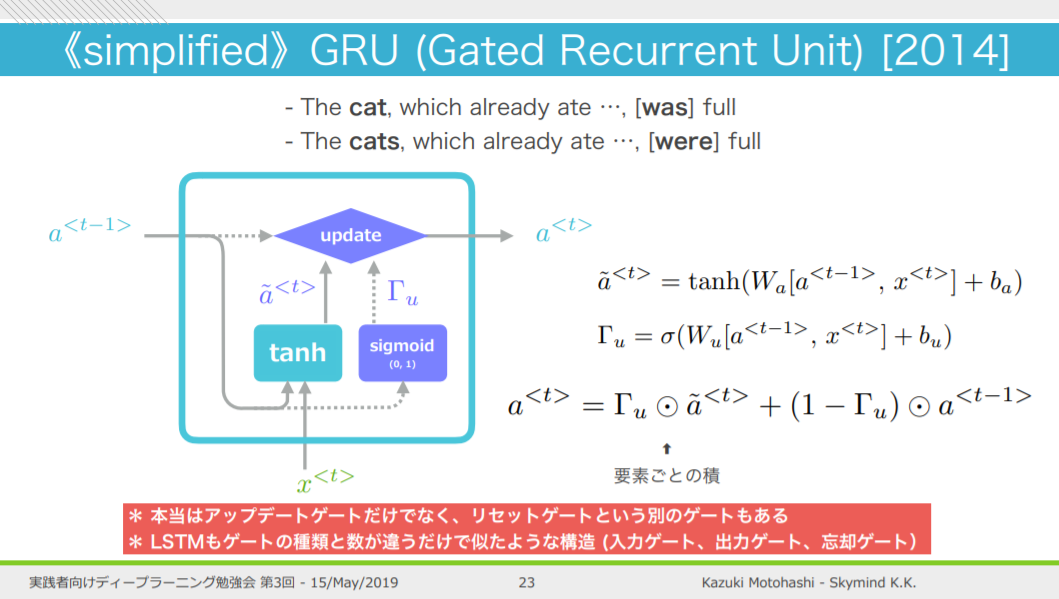


図 GRUについて

GRUというのは2014年に提唱されたモデルですけど、実際には図にあるアップデートデートという情報を更新するか否かを決めるような0か1かだけでなくて、リセットゲートと呼ばれるそもそも情報を忘れる、つまり今までのメモリをリセットするという別の計算もあります。ただしこれらは概念的には同じような機能となります。RNNの仲間にはGRUの他に、LSTMと呼ばれるGRUよりもさらに有名なモデルがありますが、これもゲートの種類と数が違うだけで、ほとんど似たような構造で少し複雑ではありますがGRUと同じような考え方で理解できるものなので、式を順に追っていけば理解可能なそんなに難しい概念ではありません。

次に機械翻訳という文脈の中では、１つ強力なもので下図に示すようなseq2seqというモデルが存在します。これはシーケンス・ツー・シーケンスの略で、エンコーダ-デコーダモデルとも呼ばれています。これはどのようなイメージかと言いますと、さきほどのようなイメージと同じで、下図の四角にそれぞれRNN、GRU、LSTMなどのユニットがあって、それに対して文章を下からどんどん入力していきます。そこでエンコードされた図で示すところのCにあたる場所で文脈（コンテキスト）のような状態を次のRNN等に対してまた入力してあげて、新しい出力を得るというようなことを行っています。イメージとしては’Dogs are cute.’というような文章があったときに、それを図のCの場所で中間的な状態に押し込んで、犬はかわいいという文章を出すというような、一度文を抽象的な状態にエンコードして、それをまた日本語の文章としてデコードするというようなことを行っているモデルです。

このモデルは機械翻訳とか対話などのタスクで用いられています。すでにお気づきかもしれませんが、このようなモデルを出すと、入力の文章が増えても、ただ順番にネットワークに対して文章を入力していっているだけなので、文章の長さによらず図のユニット数のような同じ構造のモデルを用いて処理することができるので、時系列データであったり対話文であったりというような自然言語の例題に対して有用なものであるであるということが言えます。

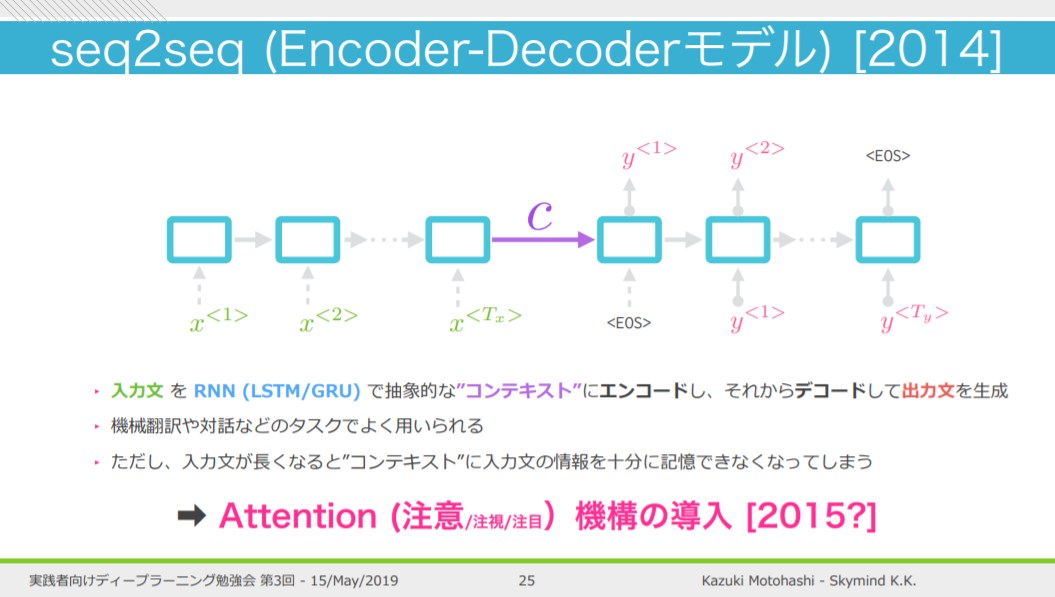
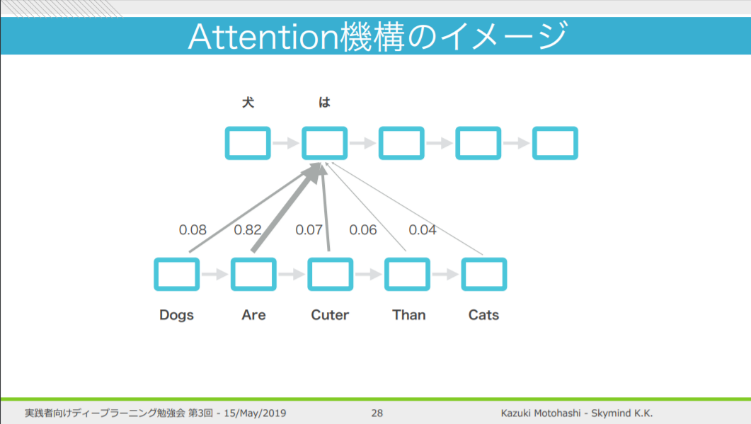
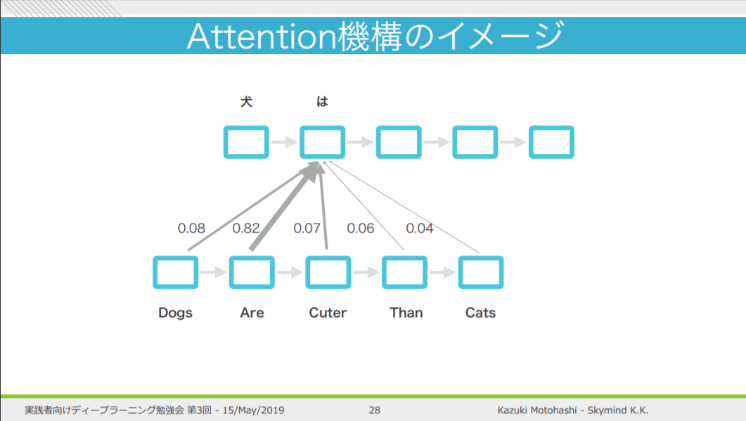


図 Seq2seqモデル

上記のモデルは今でもよく使われているものではありますが、入力文が長くなりすぎる場合、たとえば英語の長文などをこのCの場所でそれまでの状態を押し込める、つまりは大量の情報を一つのベクトル詰め込んでしまう結果、学習効率があまり良くはありません。それを解決するためにAttention（注意）機構が導入されました。

Attention機構は非常に簡単な概念で、イメージで言いますと、例えば’Dogs are cuter than cats.’みたいな文章があったときに、これを日本語に翻訳したいとします。結果として’犬は猫よりかわいい(という感じの文章になれば良いわけですが、犬という単語を出力したいときに、あまり後半の単語は気にせず’犬は’という部分に注力して見ていれば最初に出すべき日本語は犬だということが言えるわけです。つまりある程度どこに注目すべきかを重みをつけてあげるという機構がAttentionと呼ばれるものです。たとえば、次の単語の’は’を出したい場合には、’are’という単語を見れば’は’という単語が出てくるとか、下図のよう’cats’のところを見ておけば’猫’で、’than’とかがあってかつ前後に比較級’cuter’や2つの名詞が挟まっていたりしたら、何々と何々を比較して’より’というような単語になるだろうなどと考えられるわけです。また最後には’cuter’があるから’かわいい’となるというような感じで、人間が英文翻訳をするのと同じような注目の仕方でそれぞれの単語の状態を見て翻訳のタスクを行うのがAttention機構がやっていることのイメージとなります。実際、人間が翻訳するとき、’dogはcat（thanがあるから）より（cuterがあるから）かわいい’というような注目の仕方で行っていると思います。





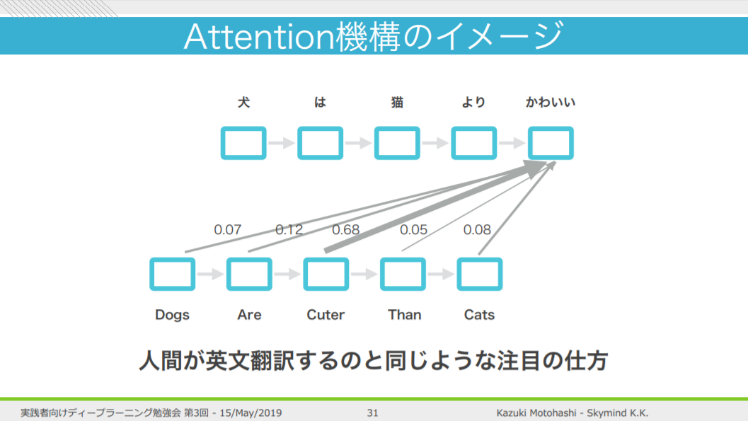


図 Attention機構のイメージ

次の図は非常に複雑ですが、どういう風に入力文の状態を注目して出力を出すかという点については1つに限定された定義が存在するわけではなくいろんな実装の仕方があります。今回はTransformerというモデルで使われているもので、内積注意と呼ばれていてる行列の内積という演算を使って各ベクトルの関連の度合いを計算するものです。

図の文章の場合には、’He loves to eat.’という文章がありますが、こちらをRNNにどんどん入力していきます。それぞれのRNNの中間層の状態は１つのベクトルになりますが、それらのベクトルと出力部分の状態の内積を取ることで2つのベクトルの関連性を調べます。ちなみに、内積というのは、掛け合わせるベクトルが同じ方向を向いていれば１で、直交になると0になり、真逆の方向を向いていれば-1になるようなそのような値を取る計算式なので、内積の大きさというものが2つのベクトルの間の関連度合い、つまりどれくらい似ているのかを表すのに用いることができます。この内積の計算をすることで、入力の状態と出力の状態の一番似ているところを図の赤の矢印のようにSoftmax関数に取り出して入力のところに掛けてあげることで、入力と出力の一番近いところを取ってきてあげるというができるようになります。

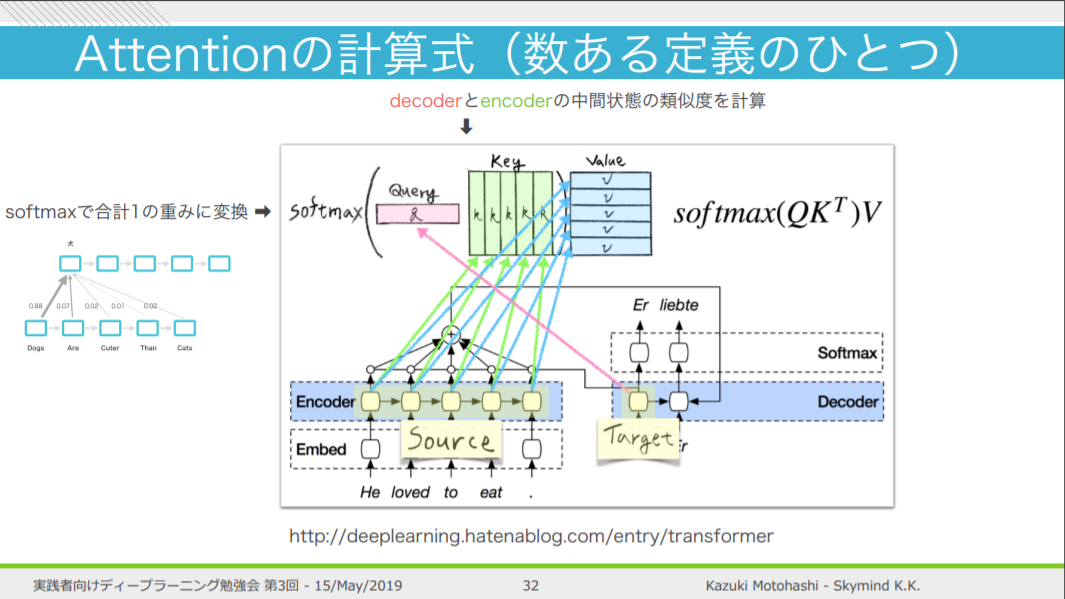


図 Attentionの計算式

今の例では、入力と出力の関連性を見ました。次の図のものはもう少し複雑な説明にはなりますが、１つのベクトルに対して同じ計算を行っていくと、同じ文章の中でどの単語とどの単語が関連し合っているかを学習することができるようになります。この技術はSelf-Attention（自己注意）と呼ばれています。こちらの図はもともとミクシーの方によって書かれたものを利用させていただいていますが、たとえば、好きな動物はというような文章があったときに、このSelf-Attentionを用いて学習すると、図の〇の大きさが関連性の高さを示しているのですが、’動物’という単語と’好き’という単語の関連性が高いというような学習結果が得られます。’動物’という名詞がどういう動詞に関連性があるか、つまり単語がどの単語を修飾しているかといったことがSelf-Attentionを使うと判定できます。

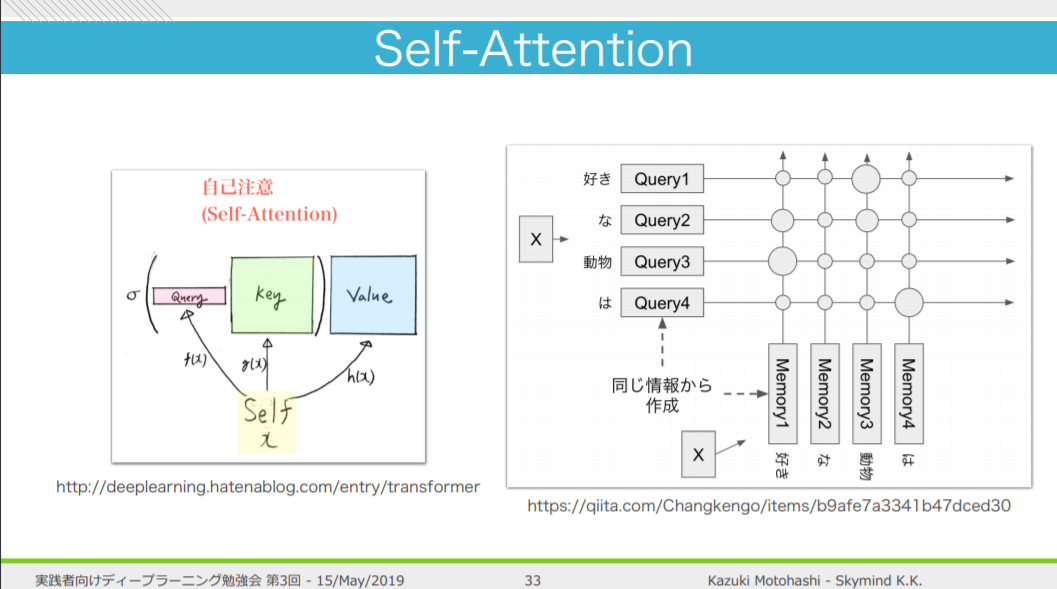


図 Self-Attentionとは

次の図は、TransformerというGoogleのニューラル機械翻訳のモデルの論文から取ってきたものになりますが、’The law will never … ‘というような文章があった場合に、図の’its’が何を指しているのかを学習することができます。’its’は代名詞なので、その前にすでに具体的な単語があってそれを指し示しているのですが、それはこの場合は’law’とつながっているとか、’its application’の’its’が何を修飾しているかというようなことを確認することができます。つまり、このSelf-Attentionというものが文章の文脈の表現を抽出するのに用いることが可能であるということをここで示しています。再度説明すると、文章つながりであったり、単語同士の関係であったりといったことを抽出することがSelf-Attentionを使うことで可能になります。

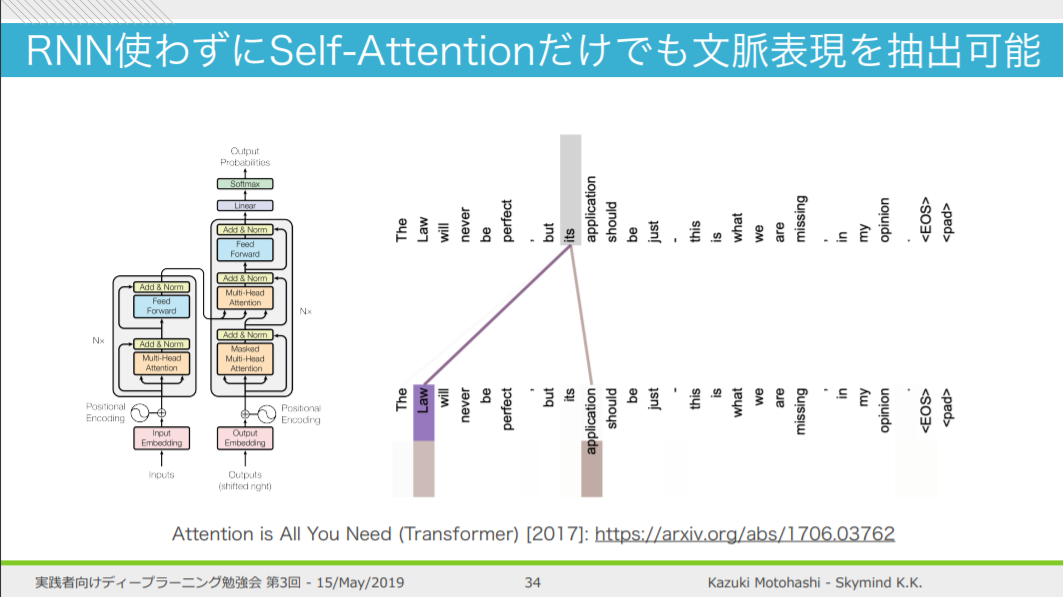


図 Self-Attentionを用いた文脈表現の抽出

ここまでの説明でようやくBERTがどういうものかということが何となくイメージできるようになったのではないかと思います。これからBERTというものをもう少し説明してきます。

BERTの詳細説明

下図がBERTのモデルの構造になりますが、いままで説明したSelf-Atrtention機構を使って文章の文脈を獲得する部分が下図の右側の図のモデルの赤の太枠で表示されている部分にあたります。下図ではtmとして示されているこのSelf-Attention機構を図の左枠の部分のようにたくさん並べます。まず文章は時系列で左側から順番に流れるようになっていますが、それだけではなくそれぞれのSelf-Attention機構には図の左枠内の矢印のように右から左向きに文章の流れとは逆方向の接続も設けることにより、さらに強力に表現を抽出できるようにした双方向の言語表現をエンコードするモデルとなっています。

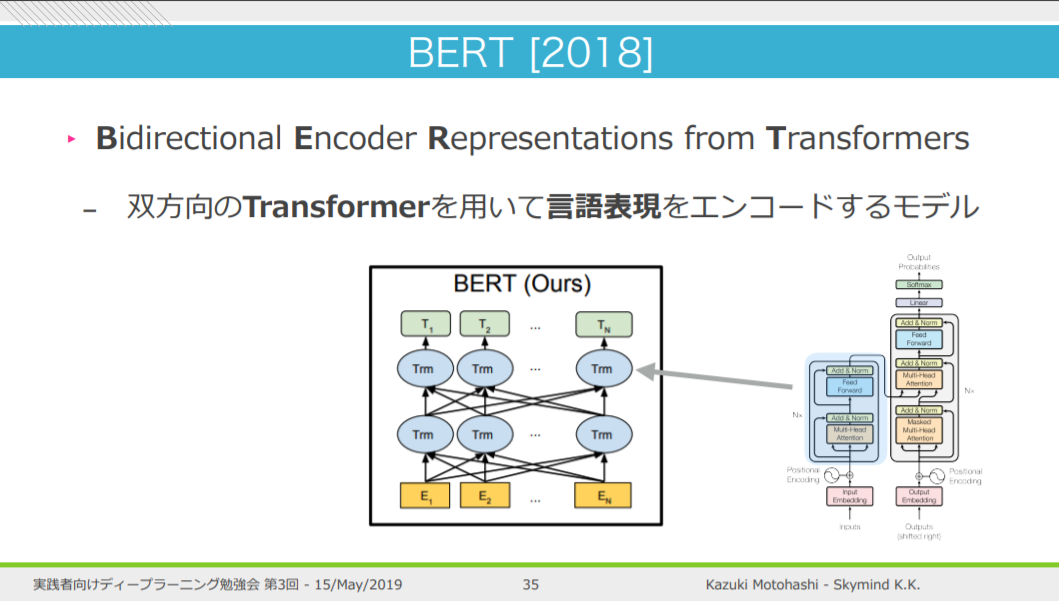


図 BERTとは

BERTを使うと何がうれしいかというと、文章の表現を学習および記述することができることです。何らかの日本語の文章で、たとえば、’神田の鬼金棒のラーメンは美味しい’のような文章が入ってきた場合に、画像の表現抽出器のように、下図の大規模データセットで訓練済みのBERTとして示されているような中間の表現抽出器を使ってある中間状態にして、それを下図のように利用することで、機械語翻訳モデル、対話モデルやこの文章がポジティブかネガティブかを調べるような感情推定モデルなどを作ったり、いろんなタスクに応用できたりします。ありがたいことに、京都大学などでWikipediaなど大規模な文章のデータセットを使って、BERTのモデルの訓練を行ってくれています。Googleが提供しているモデルには、英語、中国語など多言語対応しているものもあれば、日本語に対応したものもあります。いろんな人がやりたいようなタスクの性能をあげるためのひとつのマリオのスパワーアップキノコのような存在としてBERTを使うことができます。



図 BERTの表現抽出器としての利用

以下の図はBERTが多数のベンチマークタスクの性能のリストで圧倒的な高い性能を発揮したということで話題になりました。State-Of-The-ArtとはAIのベンチマークの世界で最高性能のことを指します。たとえばSWAGというベンチマークテストはどういったものかというと、’A girl is going across a set up monkey bars. She’のような文章があった場合、それに続く文章を当てるというようなタスクになっています。選択肢としては図に示されるように4つのものがありますが、この中で一番自然なつながりがどれなのかを選ぶというものです。これはなかなか人間でも少し時間をかけて考えることで理解できるような難しい問題設定ですが、このようなものもBERTを使うとかなり高い精度で正解することができます。このような部分がBERTがすごくて驚くべきところです。

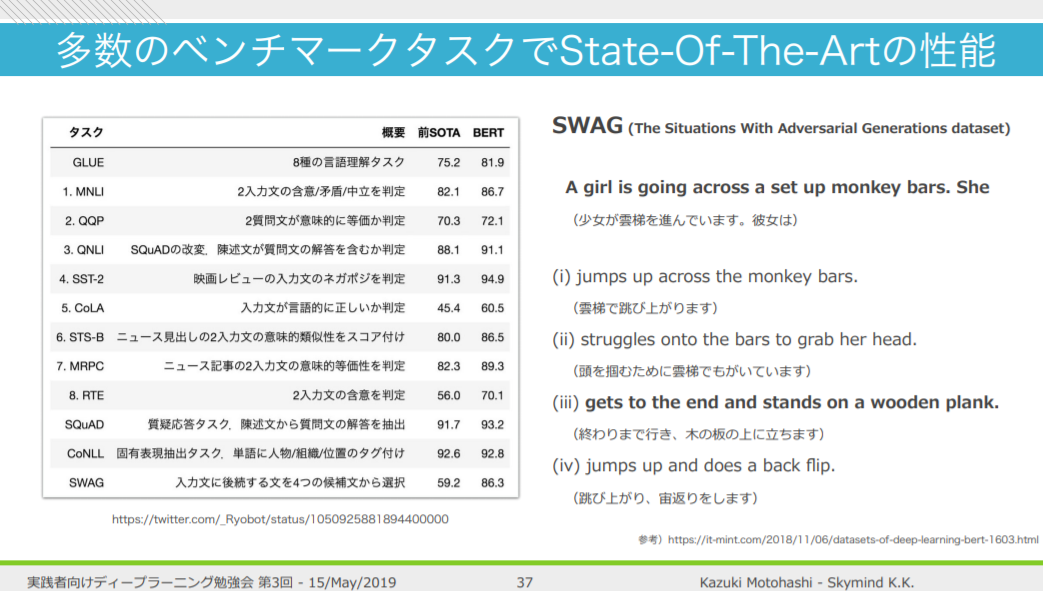


図 BERTのベンチマーク性能結果

では、BERT自体はどのように学習されているかというと、Wikipediaのような文章を入力すると勝手に学習するわけではなくて、BERTに対して何かしらのテストというか宿題のようなものを用意してあげる必要があります。やっていることは下図に示す穴埋めクイズと隣接文クイズというものを解くことで、それらをBERTに学習をさせています。これはどういうものかというと、’the men went to grocery store’ というような文章があって、図のようにその中の１つの単語、この場合はgroceryの部分をマスクすることで、中にはどのような単語があったでしょうかというようなことを学ばせることをやっていたり、さきほどのSWAGのタスクと似た感じで、入力文章から次の文章がどれかを当てさせたりするようなものです。図の例の場合では、上の文章が正解なのですけれども、‘男がスーパーに行って1ガロンのミルクを買いました’というような文章になるため、文章として適切であると言えるので、ラベルにIsNexttというものが付いています。それ以外に図の下の文章のようにランダムに別のところから文章を取ってきますが、たいていランダムにとってきた場合には前後の文章としては意味が成立しないので、ラベルとしてはNotNextというものが付いてくるわけですが、実際2番目の文章には“ペンギンは鳥より飛ばない”というように隣接する文章として正しいか否かを当てるようなタスクもあります。このようなことを学習していくことで、文脈にあたる表現を獲得できるようになっていきます。

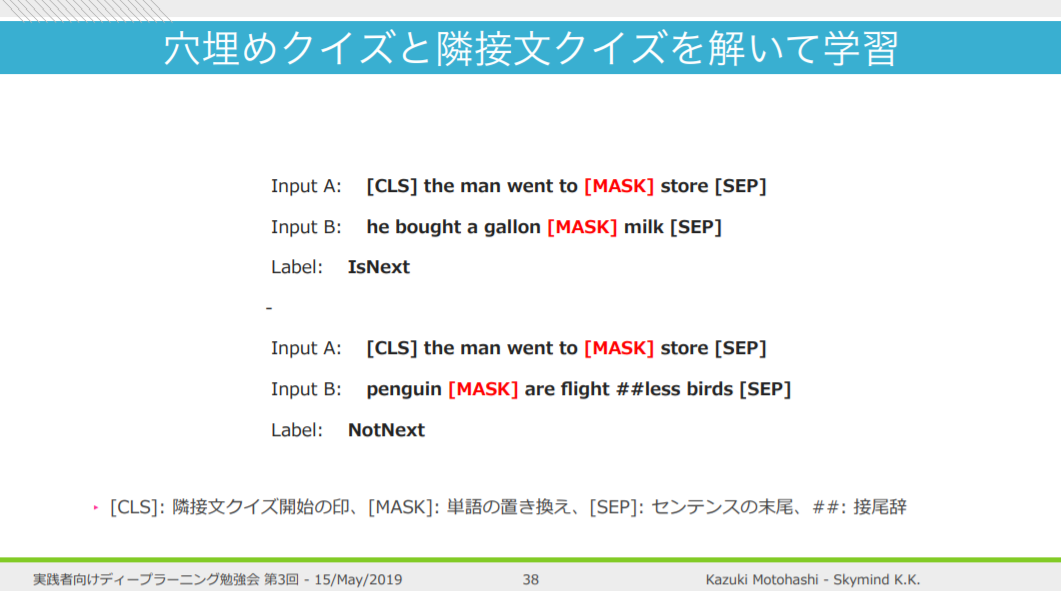


図 BERTの学習パターン

このようにして獲得したBERTのモデルに対していろいろな出力のユニットがありますが、それにアドオンでネットワークを追加してあげれば、さまざまな機能を実現でき多様なタスクに応用できるようになります。たとえば、下図の赤の点のように、分類タスク用のネットワークを追加すれば、分類ができるようになるし、文章を出力するようなネットワークを後ろに追加してあげれば翻訳などができるようになります。

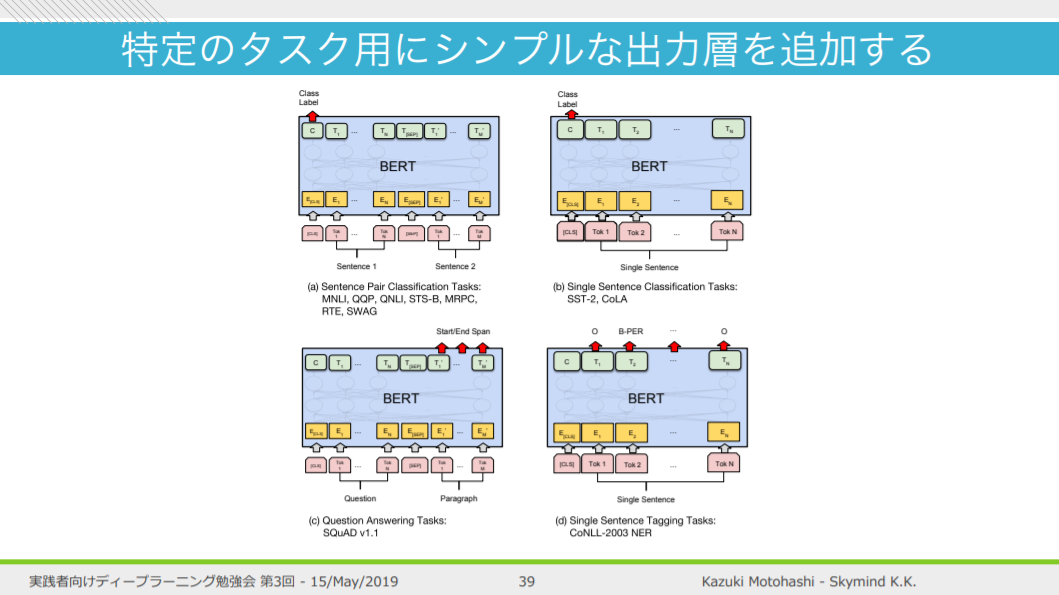


図 BERTモデルの応用の実現方法

まとめ

まとめとしては、まず時系列データに対しては、全結合ニューラルネットワークよりリカーレントニューラルネットワークの方が向いています。またリカーレントネットワークもゲートなどを追加することで単純なリカーレントニューラルネットワークの欠点を解消するこができるようになります。リカーレントネットワークの有名なモデルとしては、LSTMやGRUなどあります。機械翻訳みたいなタスクに対しては、Seq2seqのモデルが良く使われますが、その入力の文脈情報を１つのベクトルとしてまとめるのではなくて、Attentionを使ってどの文脈に注目するかを時間の関数にするというのも近年行われています。特にSelf-Attention機構はリカーレントニューラルネットワークを使っていなくても強力な文脈表現として用いることができるものです。またBERTというのは、大量のデータセットを使って事前に学習しておいてあげれば、いろんなタスクに応用できる自然言語処理のドーピング剤のような存在であります。