卒業論文

計算問題の特徴分布に基づく類題選出による自己学習支援

Self Learning Support by Automatic Selection of Calculation Exercises based on Feature Distribution of Exercises

成蹊大学理工学部情報科学科

宮地 雄也

要旨

本研究では自然言語処理の技術を人工言語の数式に適応し，分類ができるかどうかを目的とした．またその評価指標として累代選出も行い実際まちがえているのかどうかも検討した．分散表現で文字の特徴量を抽出したのちその特徴量を用いて数式の分散表現化も行い，式のベクトルを得る．この結果から現在では間違った問題から人をクループ化していたシステムから問題をグループ化してその問題を間違えた人の最適化学習支援を行い，より繊細で効果的な学習が期待できる．この研究の先には大量の個人情報を扱うアダプティブラーニングにおいて問題から分類できるようになれば，学習データが少なくても十分に効果を発揮することが期待できる．

目次

第 1 章

序論

昨今，小・中学生の理系離れが問題視されている．平成 30 年度全国学力・学習状況調査（全国学力テスト）の結果では平均正答率は小学校では算数 B が 51.7%，中学校数学では 47.6% とどちらも最も低く，ついで国語，理科の順で正答率が低い．小中どちらとも理系教科の習熟度が低いことを示している．この要因の一つに，数学は一つの計算方法が様々な分野に横断していくことが一度，苦手を生んでしまったらそこからの分野の理解度が下がり，次の分野での応用がきかないために連鎖的に苦手が蓄積することが原因ではないかと考えた．各単元のちょっとした積み残しが，後々，尾を引いていることが全国学力テストの結果から見て取れる．この状況を打破するには子供一人一人の苦手と向き合い，苦手と感じる前に理解していくしかない．しかしながら，生徒と向き合うべき教師の労働時間は過酷を極めており，ベネッセ教育総合研究所の調査では小中高の教員の領導時間は増加の一途を辿っていることを明らかにした．表 1.1 は文献 [3] での調査の結果の抜粋である．

表 1.1 によると，教員の労働時間は 2010 年に比べて 2016 年の方が各年次とも増加しており，教員のやることが増えている一方で，主であるはずの教材研究や教務準備に時間が避けていないことを示している．この状況では先生が生徒一人一人に時間をさき，指導することは難しい現状がつづいている．

この打開策として，IT 技術駆使した個人別最適化学習に注目が集まっている，しかし，教育の情報は，生徒の情報と結びついている個人情報なためオープン化できず，現在でているサービスでは各サービス利用者の利用状況からデータを取得し，その運用に利用しているため一部の大手企業が情報を独占している．そこで個人の統計データではなく，解く数式の方に着目し，計算式自体の特徴を抽出し，間違えた問題と同様の特徴を持つ問題が復習する類題として最適なのではないかという仮定のもと，本論文では数式の特徴を掴むために自然言語処理の分野で使用される分散表現を

表 1.1　出勤時刻・退勤時刻・学校にいる時間（平均時間、経年比較（教員年齢別〔公立全体〕））

	調査年	25 歳以上	26～30 歳	31～40 歳	41～50 歳	51～60 歳
出勤時間	2010	7:44	7:43	7:44	7:42	7:42
	2016	7:44	7:43	7:44	7:42	7:42
退勤時間	2010	19:30	19:40	19:10	18:57	18:31
	2016	20:00	19:54	19:26	19:05	18:46
学校にいる時間	2010	11 時間 46 分	11 時間 57 分	11 時間 26 分	11 時間 15 分	10 時間 49 分
	2016	12 時間 26 分	12 時間 18 分	11 時間 46 分	11 時間 26 分	11 時間 06 分

適用し，さらに再帰ニューラルネットワークを用いて数式ベクトルを作り出すことを目標とし，そのベクトルを用いて実際に復習問題推定を行った．

第 2 章

分散表現

自然言語処理ではコンピューターで演算するために各単語を判別するために単語一つ一つを onehot ベクトルというものに置き換える．onehot ベクトルとはある語彙数 V の文章の中の単語 w_i がある時，次元数 V のベクトルに対し i 番目の要素のみ 1 で残りが全て 0 になっているベクトルのことである (式 2.1)．

$$\boldsymbol{A} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix} \quad (\mathrm{i} = 3 \text{ の時}) \tag{2.1}$$

これにより単語一つ一つを別々のベクトルとして区別して表記することができる．しかし onehot ベクトルには問題点があり，一つ目として，ある単語の語彙数 V が増加すると比例して onehot ベクトルの次元 V も大きくなり，処理に時間がかかる点がある．また二つ目として onehot ベクトルでは情報が 0 か 1 しかないので疎なベクトルができる．疎なベクトルとは次元数が大きくてもそのベクトルの持つ意味が薄いベクトルをさす，このような問題を解決しようと考えられたのが分散表現である．

分散表現とは疎な onehot ベクトルを密な密な実数値をの集合をベクトルとして扱い，その実数値で単語の意味を表そうとする手法である．これにより疎なベクトルであった単語のベクトルを密な表現ができ，また，密度が上がることでベクトルの次元数を削減することができる．以下，3 手法はその分散表現を得るにあたって機械学習を応用した推論をベースとして編み出された手法である．

2.1 Continuous Bag-of-Words Model

Continuous Bag-of-Words Model(以降，CBOW と略記) は文献 [1] で提唱された手法で，ある単語数 v の単語列 $w_0, w_1, w_2, \ldots, w_{v-2}, w_{v-1}, w_v$ がある時，単語 w_t が文脈中で前後 m 単語をコンテキスト C とする時の前後 m 単語が共に共起する事前確率が増大するように学習するモデルである（図 2.2）.

図 2.1 より w_t をターゲットにした時の周辺単語 m 個を対象としたときの事後確率は式 2.2 のように書くことができる．CBOW モデルでは式 2.2 の最大化問題と見ることができる．そこで CBOW の損失関数は，単に式 2.2 の確率に対数を取りマイナスをつけ，最小化問題として解く．式 2.3 はコーパス全体に拡張した形を示す.

$$[H] \;\; P(w_t|w_{t-m}, ..., w_{t+m}) \tag{2.2}$$

$$[tbh] \;\; L = -\frac{1}{T}\sum_{t=0}^{T} \log P(w_t|w_{t-m}, ..., w_{t+m}) \tag{2.3}$$

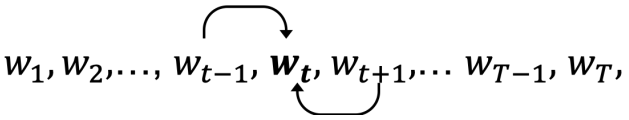

図 2.1　単語の列からターゲットとなる単語を推測する ($m = 1$)

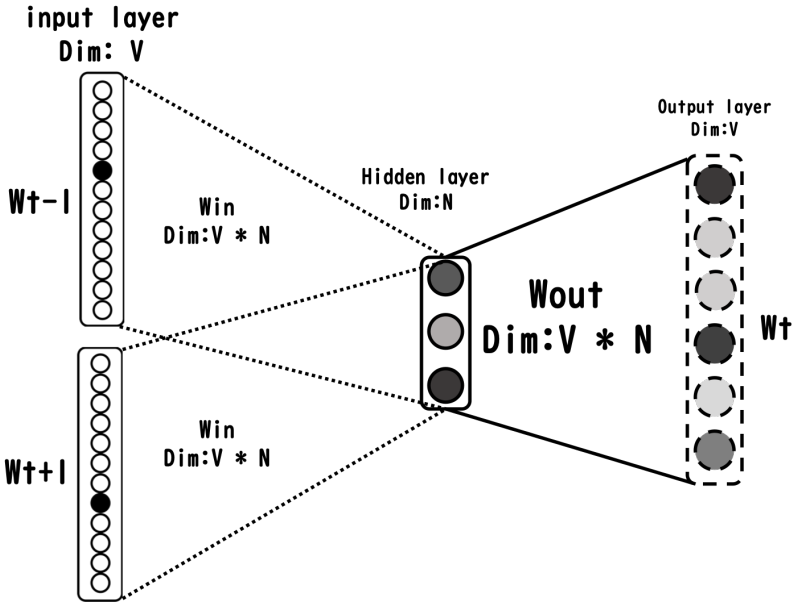

図 2.2　CBOW のネットワーク構成モデル模式図 ($m = 1$)

2.2 Skip Gram

Skip Gram は文献 [1] で紹介されている章 2.1 で述べた CBOW とは別の手法である．CBOW とは逆に中心の単語 w_t から m 個の周辺単語を推測するモデルである．ネットワーク構成は図 2.4 のようになる．

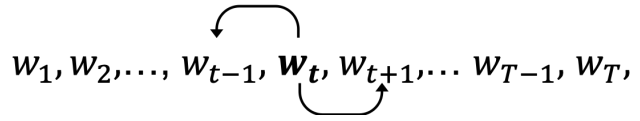

図 2.3　単語の列からターゲットとなる単語を推測する ($m = 1$)

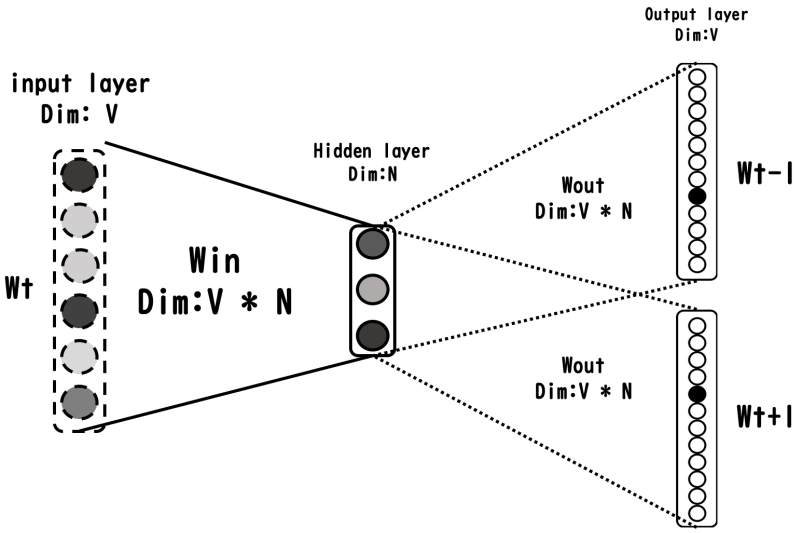

図 2.4　SkipGram のネットワーク構成モデル模式図 ($m = 1$)

ある単語 w_t が入力層に入力され，周辺単語の $w_{t-m}, ..., w_{t+m}$ を推測する時，その全てが同時に起こる確率は式 2.4 となる．

$$P(w_{t-m}, ..., w_{t+m}|w_t) \tag{2.4}$$

ここで SkipGram モデルは $w_{t-m}, ..., w_{t+m}$ のそれぞれのあいだに関係性がないと仮定し，交差エントロピーを用いて損失関数 (式 2.5) を定義する．

$$\begin{aligned} \log P(w_{t-m}, ..., w_{t+m}|w_t) &= -\log \prod_{k=0}^{m} P(w_{t-k}|w_t) \\ &= -\sum_{k=0}^{m} \log P(w_{t-k}|w_t) \end{aligned} \tag{2.5}$$

そして式 2.5 をコンテキスト C をコーパス全体に拡張すると式 2.6 となり，この式を学習によって最小化していく．

$$\begin{aligned} \sum_{C} \log P(w_{t-m}, ..., w_{t+m}|w_t) &= \sum_{C} -\log \prod_{k=0}^{m} P(w_{t-k}|w_t) \\ &= -\sum_{C} \sum_{k=0}^{m} \log P(w_{t-k}|w_t) \end{aligned} \tag{2.6}$$

第 3 章

LSTM(Long short-term memory)

順伝播ニューラルネットワークでは前の情報はつかわないため文章や音声など，一つ前の情報に影響を受けるデータに対しては有用ではない．そこで，ある時刻 t の出力の際，過去の情報も扱う再帰ニューラルネットワーク（以降 RNN と省略）というものが発案された．これは時刻 t の入力を x_t とする時，各ネットワークは過去の情報を記憶しているようにみなすことができる．しかしながら通常の RNN の場合，時間が経つほどに前の情報は薄れていく勾配消失が起こることがあった．これを解決しようとしたのが記憶情報ごとにメモリーの役割を果たすネットワークを分けた Long short-term memory（以降，LSTM と省略）というモデルである．LSTM では前の時刻 $t-1$ の C_{t-1}，h_{t-1}，x_t が入力ベクトルとして受け取る．C_{t-1} は LSTM で導入された時刻 0 から t-1 までの情報を記録しているベクトル，h_{t-1} は一時刻前の出力ベクトル，x_t は時刻 t の入力ベクトルである．それぞれのセルを計算するために 4 つのゲートを導入する．それぞれのゲートは入力セルの情報をどれくらい加味するかを重み付けするゲートである．様々なバージョンがある中でもっとも基本的なネットワーク構成を模式図を図 3.1 に示す．以下各ゲートに対して説明する．

3.1 forget ゲート

forget ゲートは過去の情報が詰まっている C セルをどの程度次の今の時刻に使うかを決める重みである．その重みは入力と一時刻前の隠れ層の出力が使われており，その式を 3.1 に示す

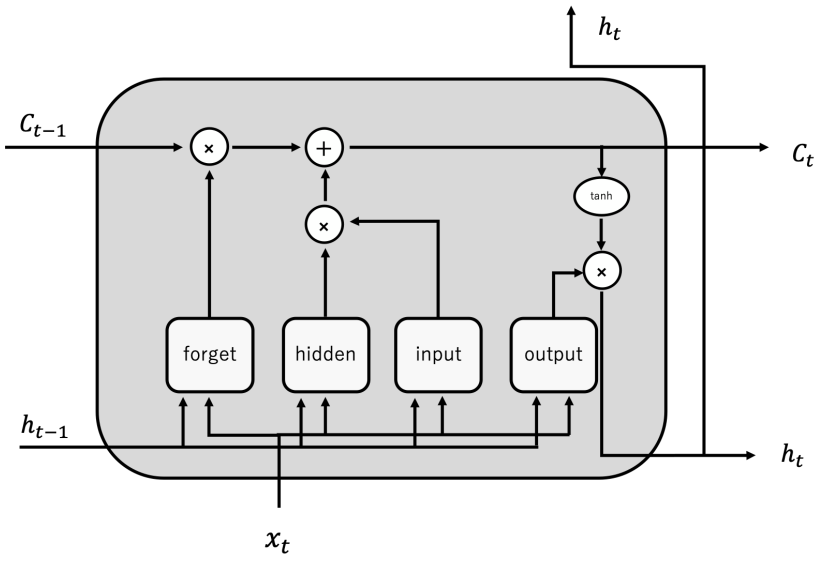

図 3.1　LSTM のネットワーク構成モデル模式図：図中の forget,hidden,input output はそれぞれゲートが扱う情報を示している

$$f = sigmoid(x_t W_x^{(f)} + h_{t-1} W_h^{(f)} + b^{(f)} \tag{3.1}$$

これにより時刻 t の C_t は式 3.2 と求められる．

$$C_t = f \odot C_{t-1} \tag{3.2}$$

3.2 output ゲート

ouput ゲートでは次の時刻にどれくらい現在の情報を渡すかを決める重みである．その重みは入力と一時刻前の隠れ層の出力が使われており，その式を 3.3 に示す

$$o = sigmoid(x_t W_x^{(o)} + h_{t-1} W_h^{(o)} + b^{(o)} \tag{3.3}$$

これにより時刻 t の C_t は式 3.4 と求められる．

$$C_t = o \odot \tanh(C_t) \tag{3.4}$$

3.3 hidden ゲート

hidden ゲートでは forget ゲートで差し引かれた記憶セル C に新たな情報を加えるゲートである．

その情報は入力と一時刻前の隠れ層の出力を用いて算出する．（式 3.5)

$$\bar{h} = \tanh(x_t W_x^{(h)} + h_{t-1} W_h^{(h)} + b^{(h)} \tag{3.5}$$

この $\bar{h}$ が一時刻前の C_{t-1} に加算されることで新たな記憶を追加する．

3.4 input ゲート

input ゲートでは hidden ゲートで追加しようとした新たな記憶をどのくらい加算するかを決める重みを求めるゲートである．その重みは入力と一時刻前の隠れ層の出力が使われており，その式を 3.6 に示す

$$i = sigmoid(x_t W_x^{(i)} + h_{t-1} W_h^{(i)} + b^{(i)} \tag{3.6}$$

これにより時刻 t の C_t は式 3.7 と求められる．

$$C_t = sigmoid(C_t + i) \tag{3.7}$$

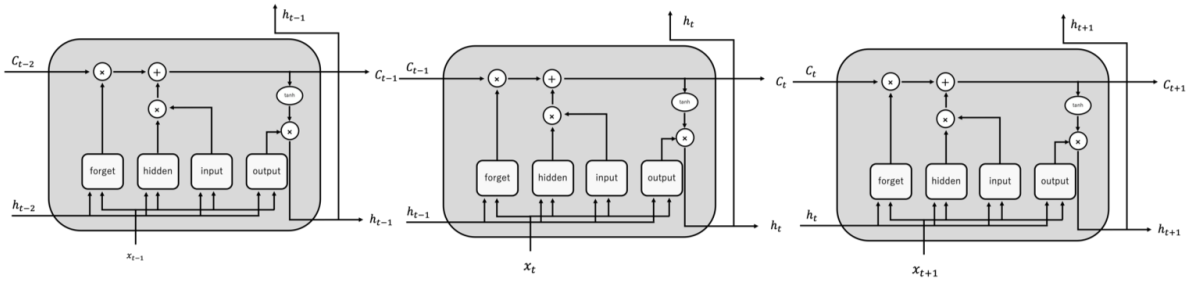

図 3.2　LSTM の 3 時刻展開:記憶を守る機構によりより高い濃度で次の時刻に情報を渡すことができる

3.5　LSTM のまとめ

章 3.1, 章 3.2, 章 3.4, 章 3.5 で紹介したものをまとめると式 3.8 のようになる.

$$
\begin{aligned}
f &= sigmoid(x_t W_x^{(f)} + h_{t-1} W_h^{(f)} + b^{(f)} \\
o &= sigmoid(x_t W_x^{(o)} + h_{t-1} W_h^{(o)} + b^{(o)} \\
\bar{h} &= \tanh(x_t W_x^{(h)} + h_{t-1} W_h^{(h)} + b^{(h)} \\
i &= sigmoid(x_t W_x^{(i)} + h_{t-1} W_h^{(i)} + b^{(i)}
\end{aligned}
\tag{3.8}
$$

$$
c_t = f \odot c_{t-1} + g \odot i \tag{3.9}
$$

$$
h_t = o \odot \tanh(c_t) \tag{3.10}
$$

LSTM モデルを 3 時刻分展開したものを図 3.2 に示す.

第 4 章

系列変換

系列変換とは再帰ニューラルネットワークを用いて時系列データを時系列データに変換することをさす．モデルの構成図の例を図 4.1 にしめす．

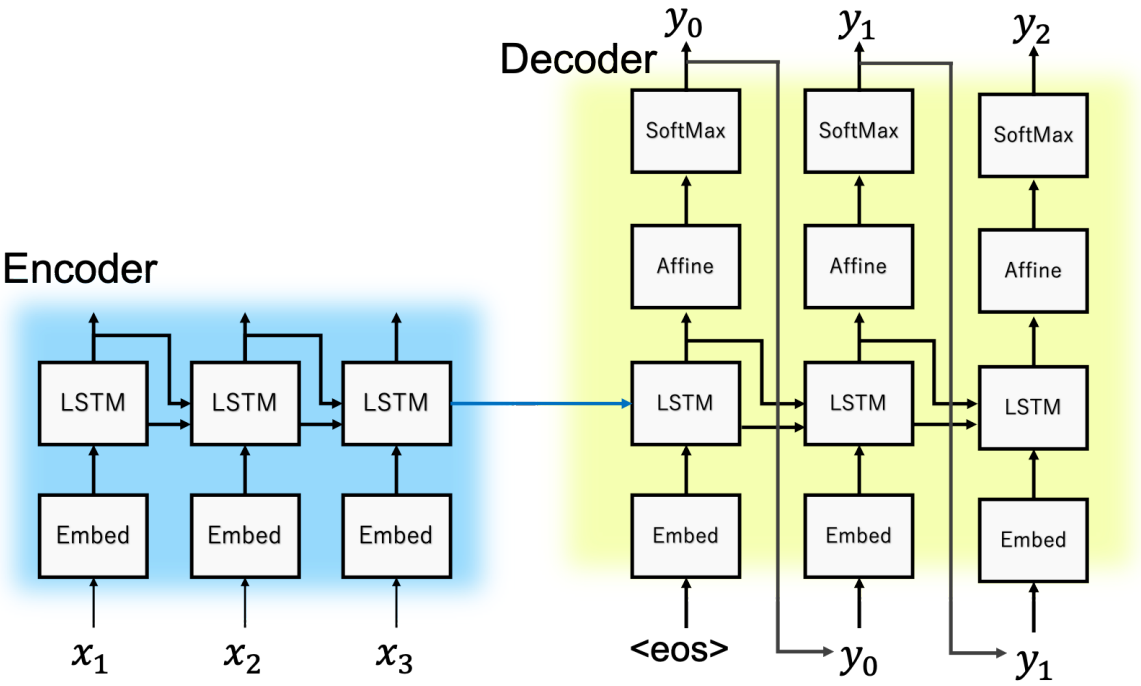

図 4.1　LSTM を用いた基本的な系列変換モデルの 3 時刻展開

大きく分けて Encoder と Decoder に分かれる．Encoder 側は入力を受け取り Embedding 層で入力系列 x_i を埋め込み行列に変換し，LSTM に入力として渡す．Encoder では LSTM が時系列データを記憶するメモリーのように振る舞い次の時刻へ情報を渡していく，時刻 t の時の LSTM の隠れ層の出力 h_t には LSTM が渡してきた $1 \sim t$ までの情報が溜め込まれており，入力系列を変換するために必要な情報がエンコードされて任意の固定長ベクトルに溜め込まれる．この固定長ベクトル Decoder 側の最初の LSTM の前の時刻の h として入力され，区切れ文字¡eos¿を入力として受け取り，エンコーダー側と同じように埋め込み行列に変換し LSTM で再帰的に学習，そして Affine 層で入力ベクトルの大きさと同サイズに圧縮し，SoftMax 層（式 4.1）でベクトルの値を確率に変換する．

$$softmax(x_i) = \frac{e^{x_i}}{\sum e^{x_i}} \tag{4.1}$$

区切れ文字¡EOS¿によって出力された y_0 を次の時刻の入力とし，2 時刻目として y_1 を出力し再び区切れ文字を出力するまで繰り返す，この技術を応用して機械翻訳など時系列データの変換に用いる．

第 5 章

Exericises Vecter Representation

5.1　システム全体の流れ

数式の特徴を取り出し S 次元のベクトルに変換し，その分布から類題選出を行うシステムを作成し概要を図 5.1 に示す，以下，各システムの箇所ごとに説明する．

5.2　計算式の特徴量抽出

5.2.1　概要

数式を分布化する際，そのベクトルの中に数式の特徴を入れ込んだベクトルを生成する手法が確立していない．そこで本論文では数式の各文字，記号を単語のようにみなし，onehot ベクトルを作成し，それを埋め込み層で特徴を踏まえた低次元ベクトルに変換したのち，系列変換モデルで読み込むことで低次元で数式の特徴を掴んだベクトルを生成できないかと考えた．

この考えを実現するために数式は我々が目にする $2x + 3 = 5$, $\frac{3x-1}{2} + 4 = \frac{2}{5}$ ではなく，

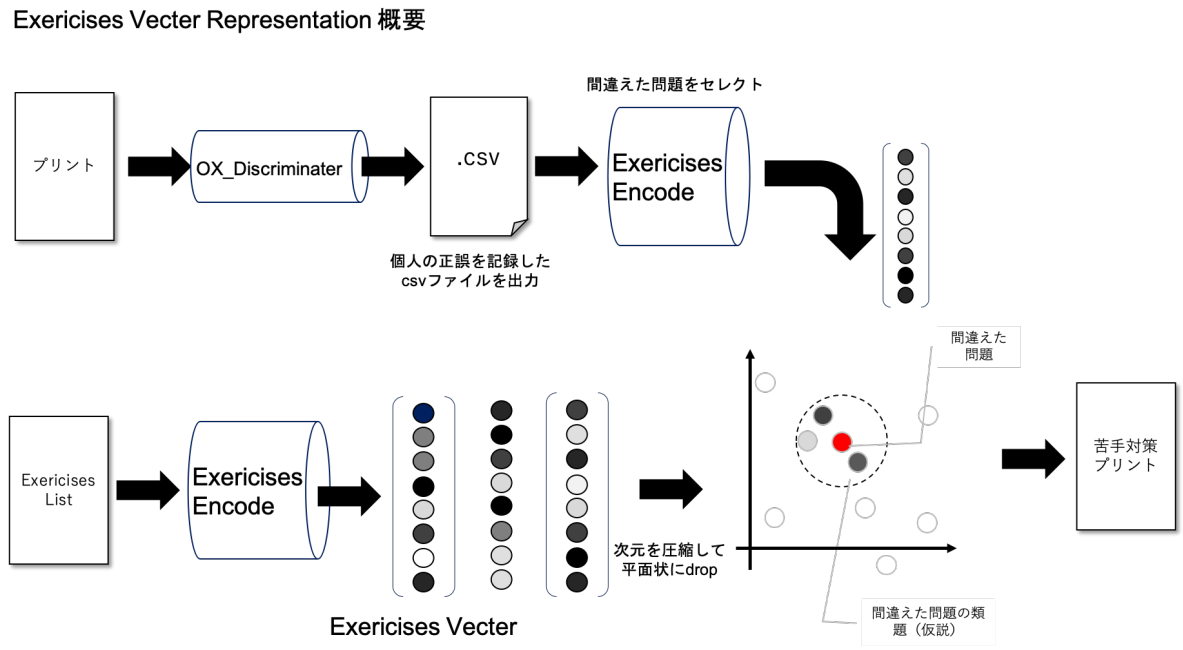

図 5.1　LSTM の 3 時刻展開:記憶を守る機構によりより高い濃度で次の時刻に情報を渡すことができる

テキスト化かつその特徴を強く受けた形に変換する必要がある．そこで本論文では数式をある一定のルールの中でテキスト化されている TEX 形式の数式を用いる．上記の計算式なら `2x+3=5,\frac{3x-1}{2}+4=\frac{2}{5}`とし，このテキストデータを用いて文字単位の埋め込んだベクトルを作成する．この式を数字は数字で,，文字，小数点，符号，丸括弧，波括弧でわけそれぞれ 1-of-k 表記でベクトル化する．

実験を行った手法は以下の 2 種法で行い，それぞれ分布を python を用いて確認した．

- CBOW
- SkipGram

onehot ベクトルの置き方は以下の 2 手法を試し，それぞれ PCA と T-SNE で 2 次元に落とし plot し状況を確認する予備実験を行った．

- 出てきた数字，数式の塊を onehot を置く方法 ($[0, 1, 2, ...3.6, 0.11, ... = .+, -]$)
- 3 桁までの数字，数式に現れる記号

5.2.2　文字分布の予備実験結果

予備実験として CBOW，SkipGram で文字の特徴が得られているかを確認した．以下その結果である．

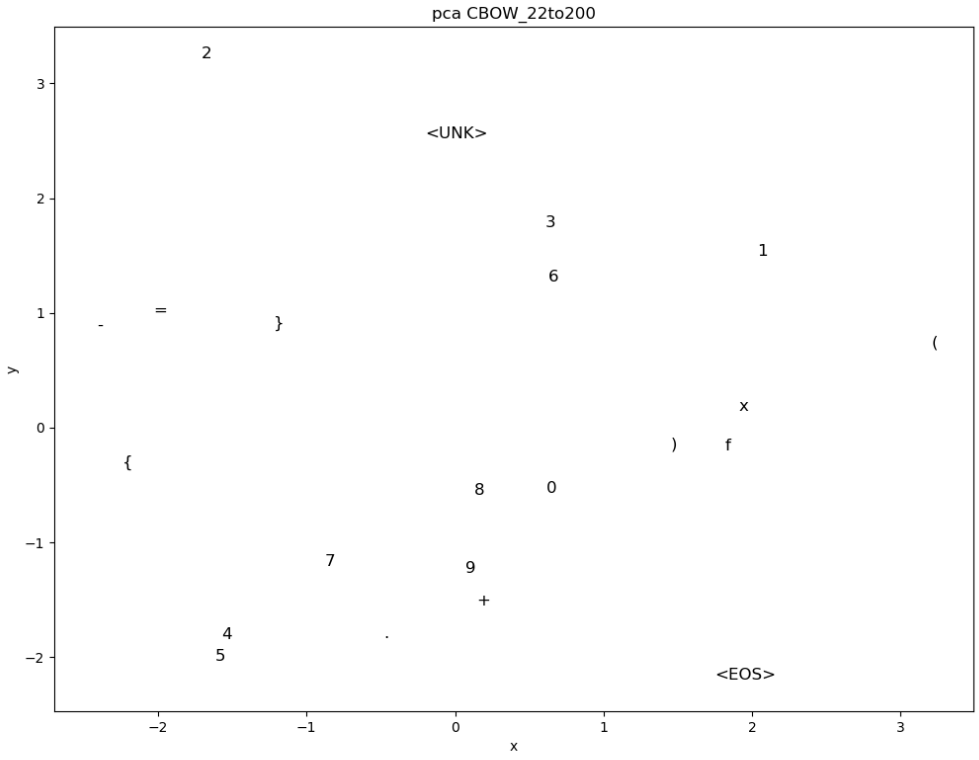

図 5.2　CBOW にて 22 次元を 200 次元に分散表現を変換したのち pca で再構成しグラフ化した結果

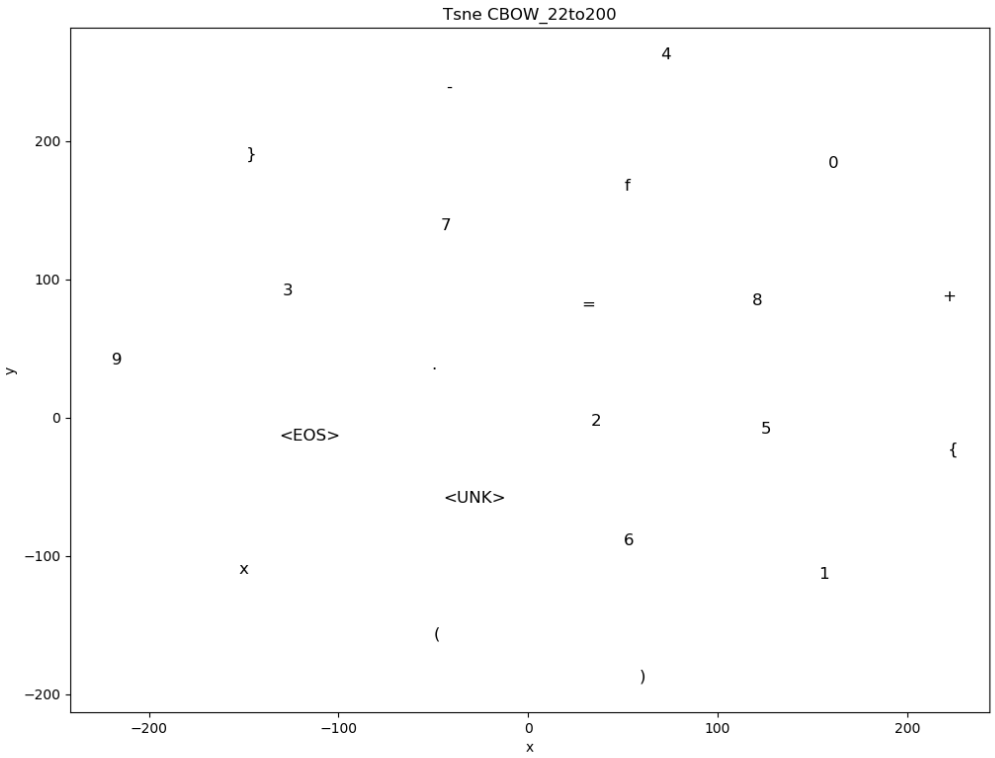

図 5.3　CBOW 22 次元を 200 次元に分散表現を変換したのち T-SNE で再構成しグラフ化した結果

図 5.2 や図 5.3 を見るとそれぞれの文字に関係はないように見える．実際にベクトルの類似度のランキングは以下のようになった.

```
input -> f
f       : 1.0
1       : 0.17398714
(       : 0.0753936
0       : 0.074519895
9       : 0.073732376

input -> (
(       : 1.0
8       : 0.09469227
f       : 0.0753936
1       : 0.041964278
<EOS>   : 0.037387587

input -> =
=       : 1.0
2       : 0.13625735
{       : 0.095088534
4       : 0.093133554
-       : 0.08290612

input -> .
.       : 1.0
<EOS>   : 0.1587607
{       : 0.13987027
4       : 0.13893604
}       : 0.08879773

input -> {
{       : 1.0
.       : 0.13987027
5       : 0.09743428
=       : 0.095088534
) >     : 0.08915341

input -> )
)       : 1.0
+       : 0.09924331
9       : 0.09857762
6       : 0.097303286
1       : 0.092290804

input -> +
+       : 1.0
4       : 0.12931207
7       : 0.12860014
)       : 0.09924331
5       : 0.06933217

input -> 0
0       : 1.0
7       : 0.1502817
<EOS>   : 0.09792947
9       : 0.084178284
-       : 0.077697314

input -> }
}       : 1.0
2       : 0.16118826
<EOS>   : 0.0995764
.       : 0.08879773
=       : 0.07274125

input -> x
x       : 1.0
3       : 0.10958019
1       : 0.09617383
<EOS>   : 0.06585052
4       : 0.04960839

input -> -
-       : 1.0
7       : 0.13888691
2       : 0.1244827
<UNK>   : 0.09517622
=       : 0.08290612

input -> 1
1       : 1.0
f       : 0.17398714
3       : 0.107833974
x       : 0.09617383
)       : 0.092290804

input -> 2
2       : 1.0
<UNK>   : 0.199631
}       : 0.16118826
=       : 0.13625735
-       : 0.1244827

input -> 3
3       : 1.0
x       : 0.10958019
1       : 0.107833974
6       : 0.09221797
4       : 0.04106327

input -> 4
4       : 1.0
5       : 0.17106494
.       : 0.13893604
+       : 0.12931207
=       : 0.093133554

input -> 5
5  : 1.0
4  : 0.17106494
9  : 0.14783658
8  : 0.101983085
{  : 0.09743428

input -> 6
6  : 1.0
<UNK>  : 0.18014975
7      : 0.100414015
)      : 0.097303286
3      : 0.09221797

input -> 7
7       : 1.0
0       : 0.1502817
-       : 0.13888691
+       : 0.12860014
6       : 0.100414015

input -> 8
8       : 1.0
9       :0.11858558
<UNK>   :0.10947606
5       : 0.101983085
(       : 0.09469227

input -> 9
9       : 1.0
5       : 0.14783658
8       : 0.11858558
)       : 0.09857762
0       : 0.084178284

input -> <EOS>
<EOS>  : 1.0
.      : 0.1587607
}      : 0.0995764
0      : 0.09792947
)      : 0.09064378

input -> <UNK>
<UNK>  : 1.0
2      : 0.199631
6      : 0.18014975
8      : 0.10947606
-      : 0.09517622
```

図 5.4　CBOW でベクトル化したのちベクトル間の類似度トップ 5 を表示

これを見てもベクトル間に関係が現れていないように見える．[2] であるようにベクトル間の演算はできないが，丸括弧 (,) と波括弧 {} のベクトル間距離を測ると

() の距離 27.04968437532474

{} の距離 27.04968437532474

このように完全に一致した．このことから word2vec を数式に用いた場合，今回の一次方程式がコーパスの際には記号の位置関係だけ学び，数式の形状は取り出せる事がわかった.

なお，ベクトル間の距離を式 5.1，類似度を式 5.2 で算出する

$$|\vec{a} - \vec{b}| = \sqrt{|\vec{a}|^2 + |\vec{b}|^2 - 2\vec{a} \cdot \vec{b}} \tag{5.1}$$

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|} \tag{5.2}$$

5.3　系列変換モデルの検討

本論文の目的としては系列変換の過程で用いれる Encoder から Decoder に渡す最終出力 h を式ベクトルとしてみなす．よって h の精度を高めていくモデルが必要となる．実験するモデルは通常の LSTM で行う系列変換，双方向 RNN を用いた bi-directionl モデル，通常 Decoder が側で利用される SkipConnection の三種類を Encoder のモデルとした．デコーダ側は通常の LSTM で行う系列変換と Attention を用いたモデルの 2 種類で確認を行うこととした．以下モデルを模式図にした．

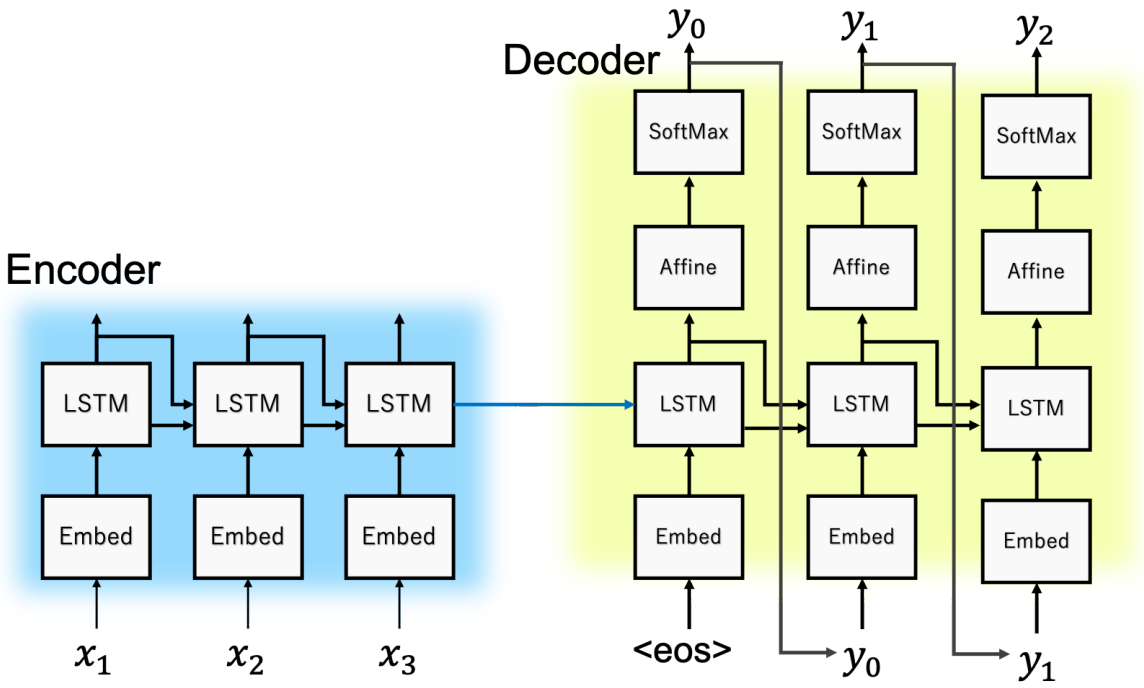

図 5.5　基本モデル

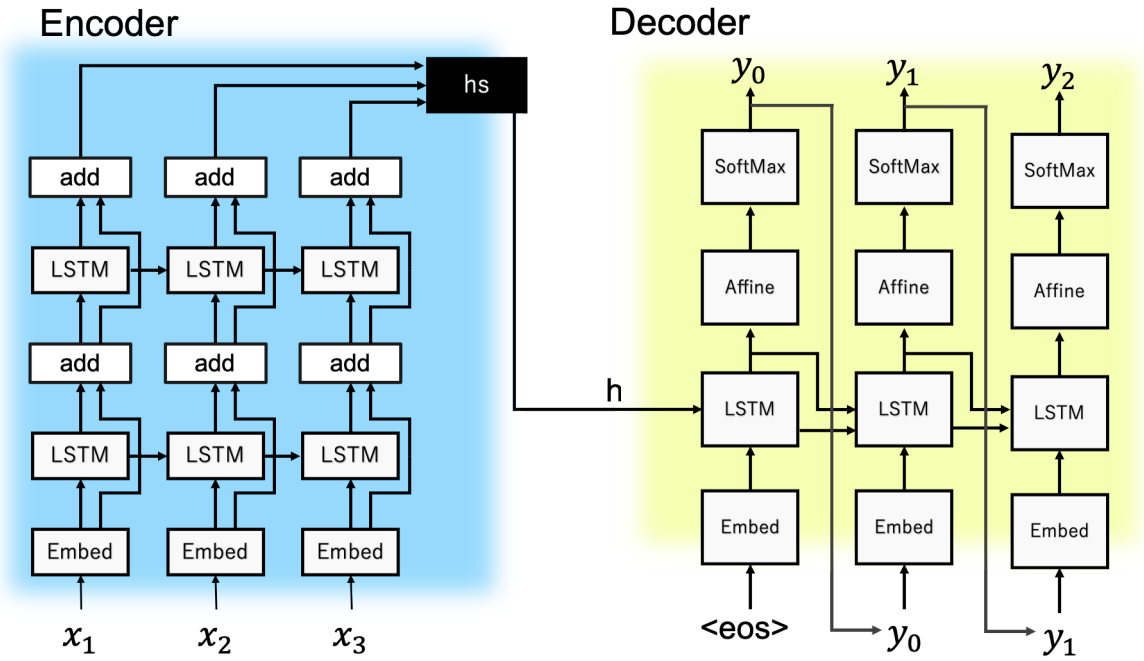

図 5.6　SkipConnection モデル

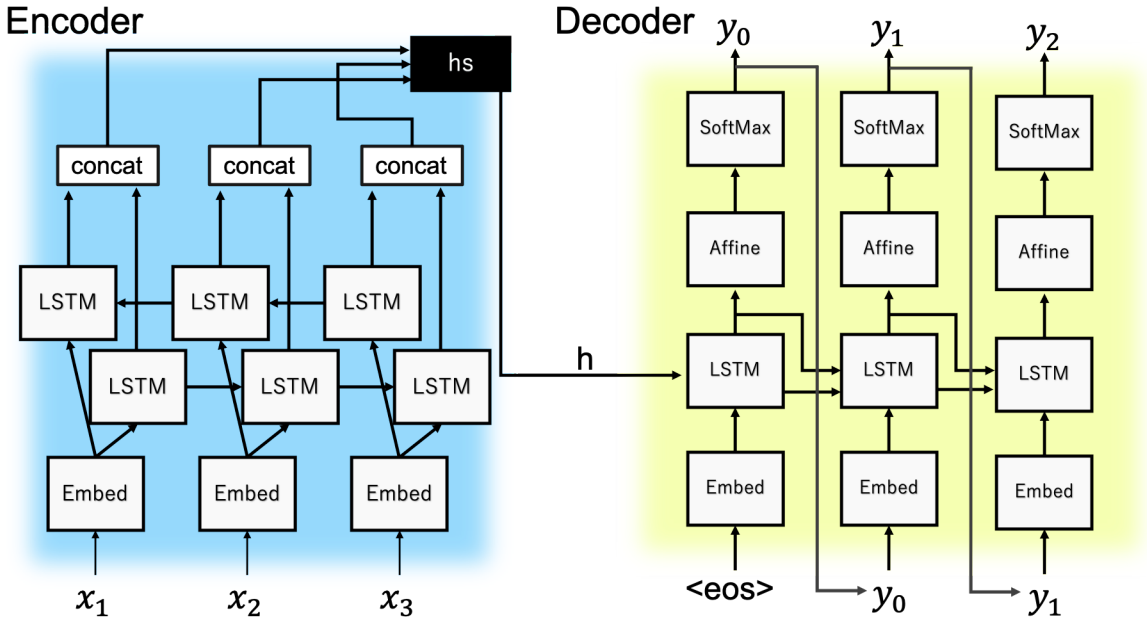

図 5.7　bi-directionl モデル

5.4　正誤判定機

5.4.1　概要

プリントから回答者の問題の正誤を判定し，記録していくのが OX Discriminaer である．正誤を判定するのには畳み込みニューラルネットを利用している．回答用紙の答えのところを切り取り，各問題に対して正解か不正解かを確認し，別途用意した，個人のプリント進行表 csv ファイルにプリント番号とそのプリントの問題ごとを正解なら o，不正解なら x としてファイルに書き込んでいく．（図 5.8）

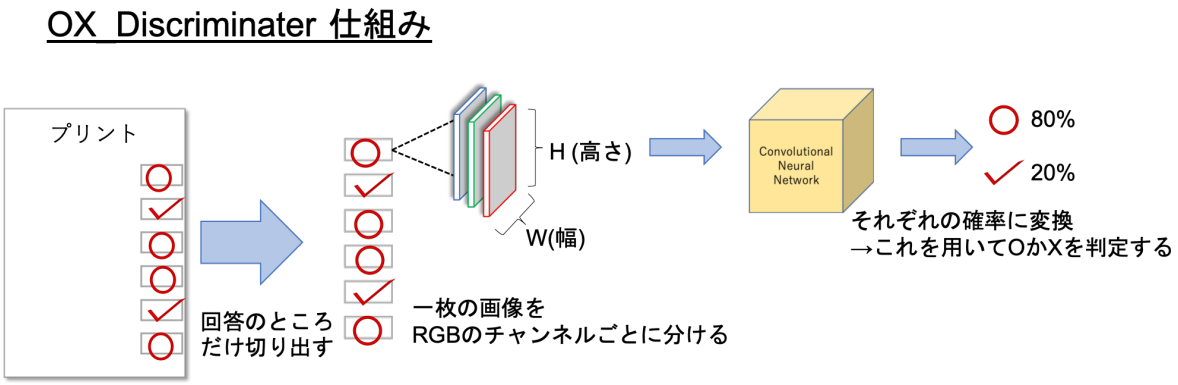

図 5.8　OX Discriminater 仕組み

この正誤を用いて間違えた問題を特定し，その数式の類題を分布に基づいて特定する．

第 6 章

結果とその検討

6.1　実験

以下の目的，方法，を元に実験を行った．

計算式の特徴量抽出

- 目的．数式のベクトル表現は可能なのかどうか
- 方法．埋め込み層，ネットワーク構成を変えながら Encoder-Decoder で復元を試みるその Encoder の出力値を数式ベクトルとしてみなし，pca,T-sne で二次元ベクトルに圧縮し図示する

実験で試したネットワーク構成を 6.1 に示す．

表 6.1　ネットワーク構成

モデル	入力サイズ	Encoder の出力サイズ	LSTM 層の数
Normal	22	200	1
		200	4
		200	8
		500	1
		500	4
		500	8
		1000	1
		1000	4
		1000	8
SkipConnect	22	200	1
		200	4
		200	8
		500	1
		500	4
		500	8
		1000	1
		1000	4
		1000	8
BiDirectional	22	200	1
		500	1
		1000	1

6.2　実験結果

以下ネットワーク**??**,**??**で学習データ 20 個を学習し，Encoder の出力の式の分散表現と思われるものの分布である．学習データの先頭 10 個を示す..

- `x+5=8`
- `5x-7=4`
- `3.6+x=-1.4`
- `3x-4=-5x+5`
- `\frac{7}{2}x+9=6`
- `3(2x-1)=9`
- `\frac{3}{7}x+2=\frac{6}{7}`
- `-\frac{4}{7}x+3=-\frac{1}{7}x`
- `\frac{x+1}{3}+\frac{2x+1}{2}=\frac{3x-4}{2}`
- `-4x+9=1`
- `5x-9=10`
- `7.2x-4=2.2x+9`
- `3(x-4)=7x-10`
- `\frac{3}{7}x-3=-5`

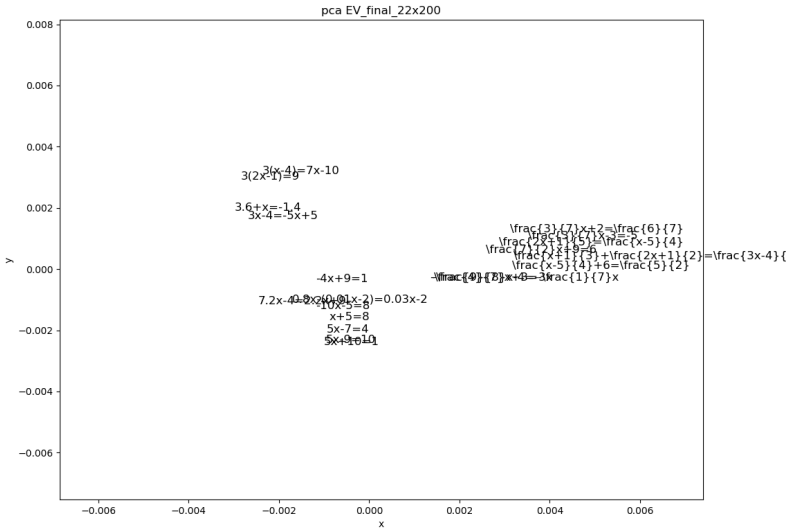

図 6.1　Normal　 LSTM 1 層 200 次元　学習データ

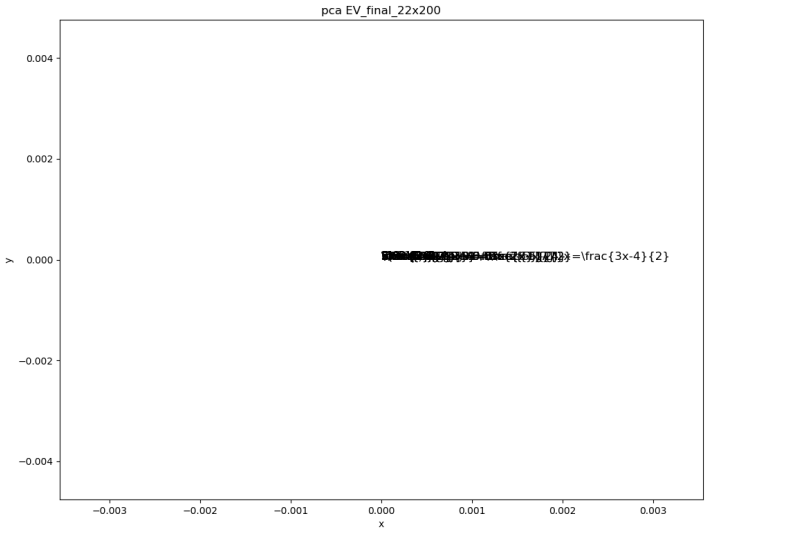

図 6.2　Normal　 LSTM4 層 200 次元　学習データ

- `-\frac{9}{8}x-4=-3x`
- `\frac{x-5}{4}+6=\frac{5}{2}`
- `\frac{2x+1}{5}=\frac{x-5}{4}`
- `0.8x-(0.01x-2)=0.03x-2`
- `-10x-5=8`
- `5x+10=1`

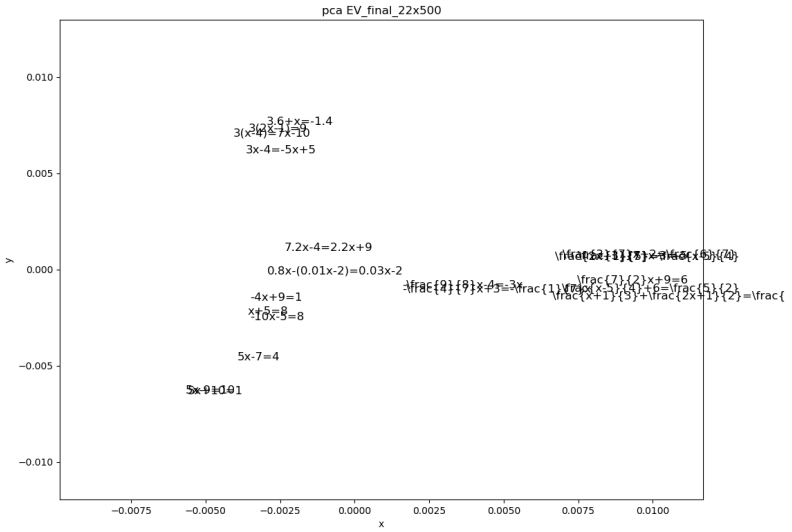

図 6.3　Normal　 LSTM 1 層 500 次元　学習データ

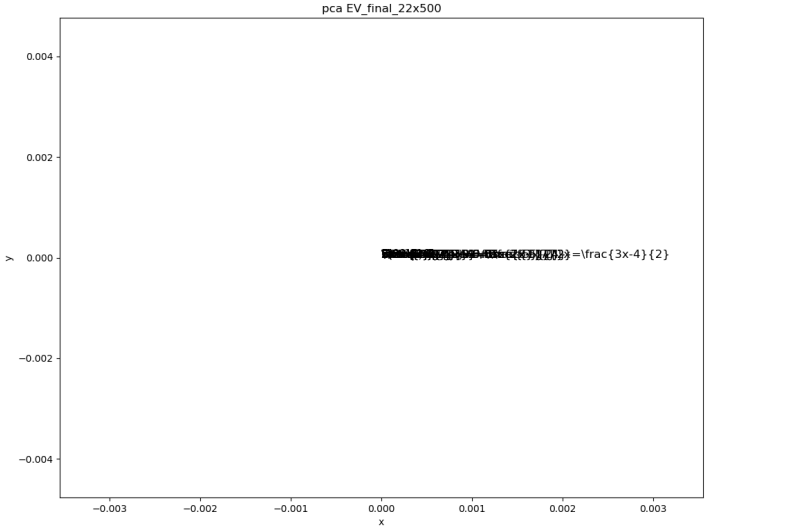

図 6.4　Normal　 LSTM4 層 500 次元　学習データ

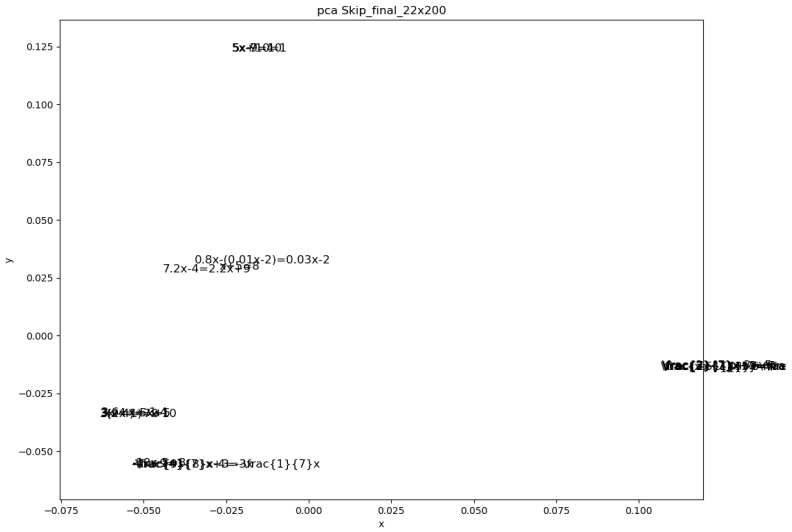

図 6.5　Skipconnect：LSTM1 層 200 次元　学習データ

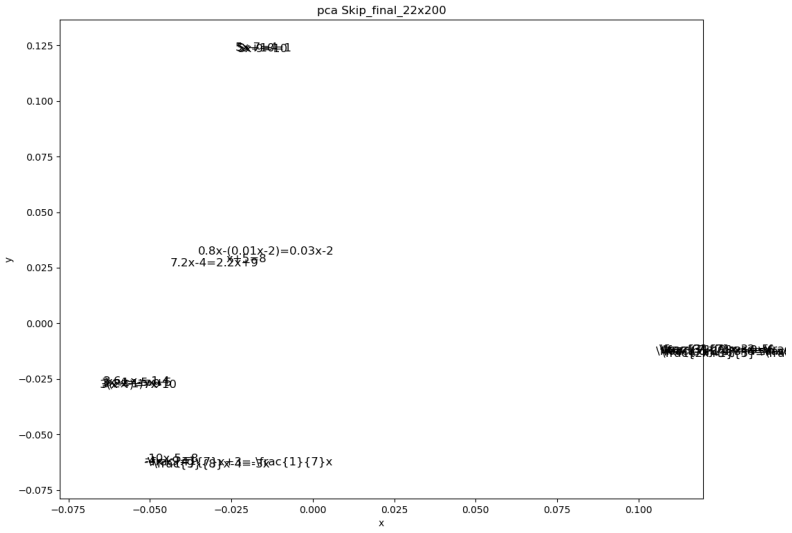

図 6.6　Skipconnect：LSTM 4 層 200 次元　学習データ

図**??**, 図 6.9，図 6.13 のどれもが同系列そうな問題が近い分布となった．特に図 6.9 は分数の中でも符号でも分かれており，高精度でベクトル化に成功しているように見える．このことより系列変換を用いて数式の分類は可能であり応用の余地がある事が確認できた．ただし，層を深くすることや，次元をあげるのでは結果が大きく向上する事がない事が判明しており，より精度の高いベクトル取得には様々なチューニングが必要だと思われる．

6.3　考察

今回の実験にて NormalEncoder，SkipConnection，BiDirectional の三種類の encoder について評価実験を行った．NormalEncoder でも 200 次元では分布として分かれていたが，LSTM を深

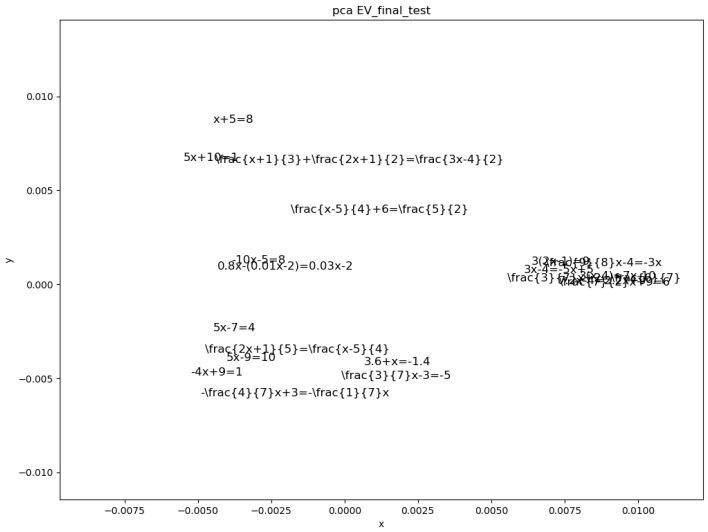

図 6.7　Skipconnect：LSTM1 層 500 次元　学習データ

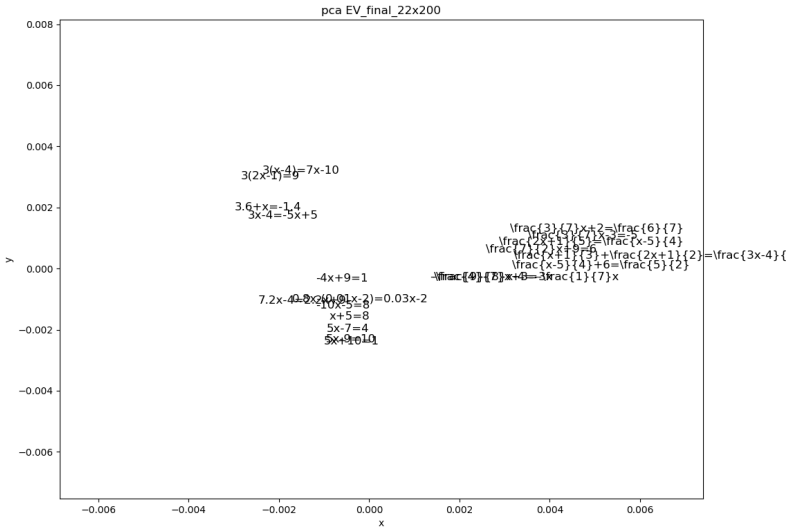

図 6.8　Skipconnect：LSTM 4 層 200 次元　学習データ

くすると勾配が消失し，一点に集中している．しかし，SkipConnection を用いると四層の LSTM を繋いでも勾配消失が防ぐ事ができる事が確認できた．

また Bi-Directional ではテストデータでも高精度に分類できており，それぞれの encoder の効果がみて取れた．

6.3.1　問題の正誤判定

研究室の同期メンバーの手書きによる訓練データ 927 件，実際の子供たちの手書きによるテストデータ 197 件にて学習を行った．正解率（図 6.14），平均二乗誤差（図 6.15）を示す．

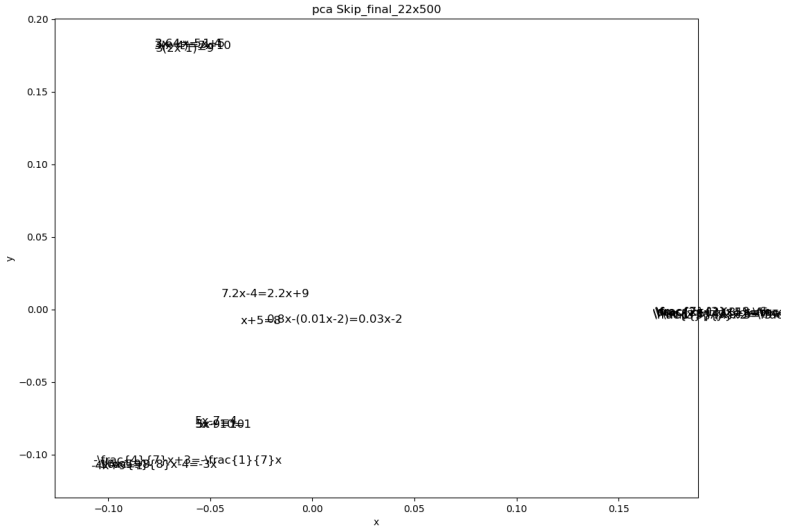

図 6.9　Skipconnect：LSTM 4 層 500 次元　学習データ

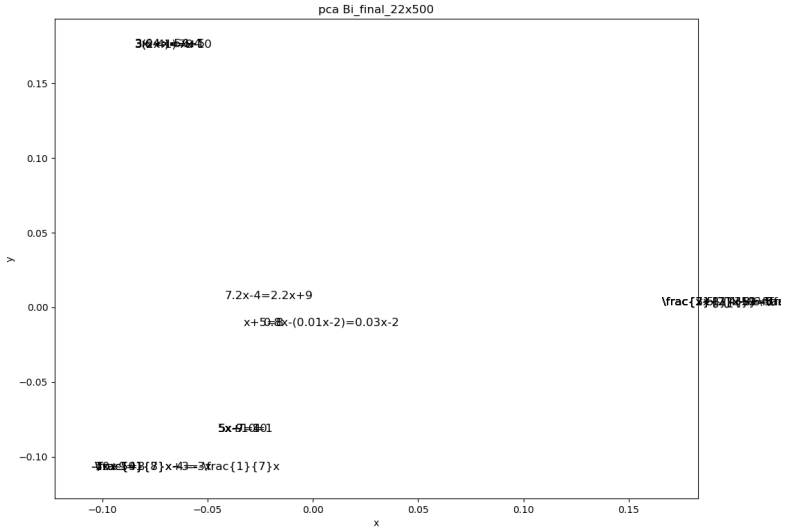

図 6.10　BiDirection：LSTM1 層 500 次元　学習データ

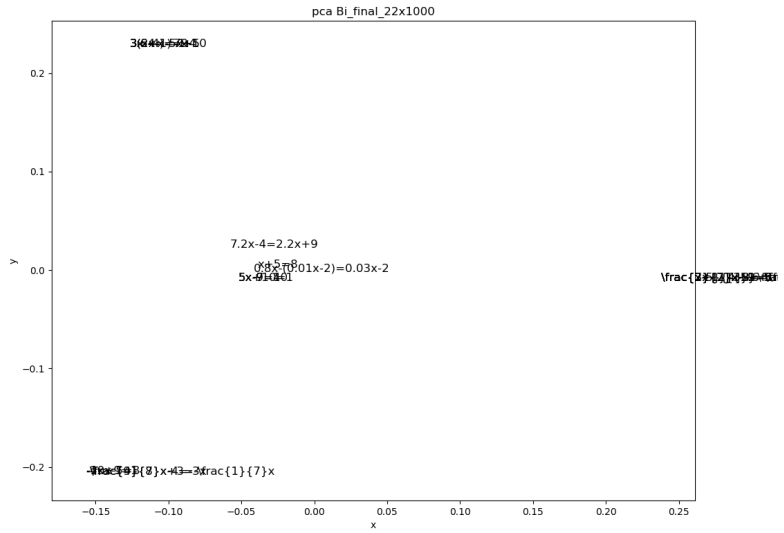

図 6.11　BiDirection：LSTM1 層 1000 次元　学習データ

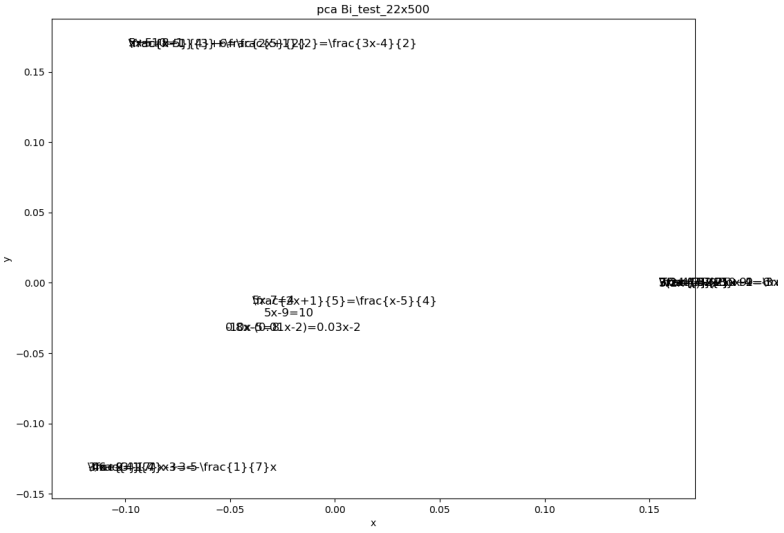

図 6.12　BiDirection：LSTM1 層 500 次元　テストデータ

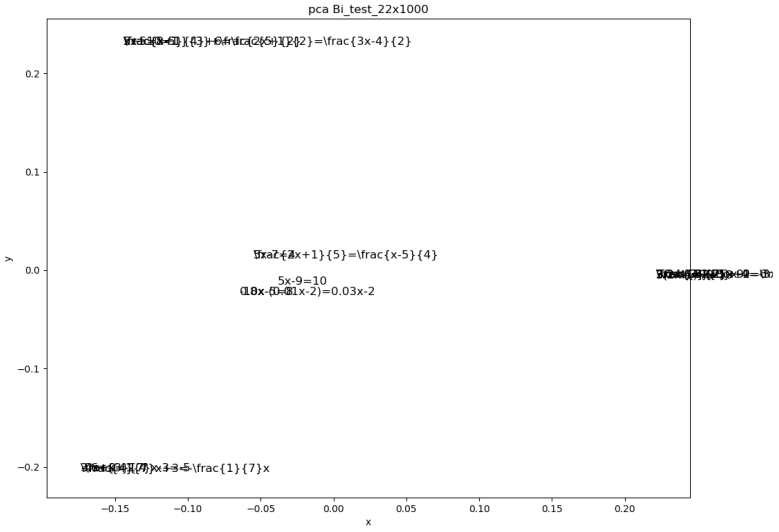

図 6.13　BiDirection：LSTM1 層 1000 次元　学習データ

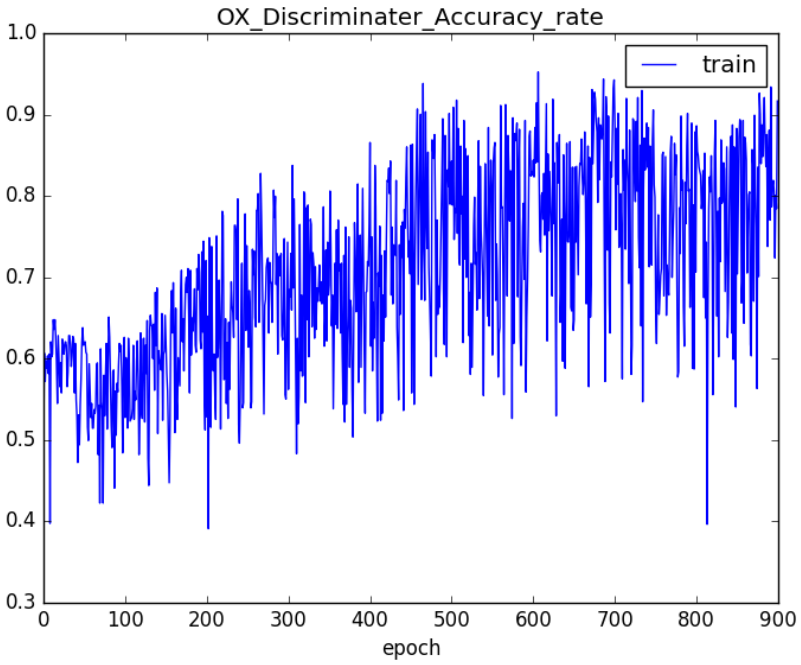

図 6.14　OX Discriminater 正解率

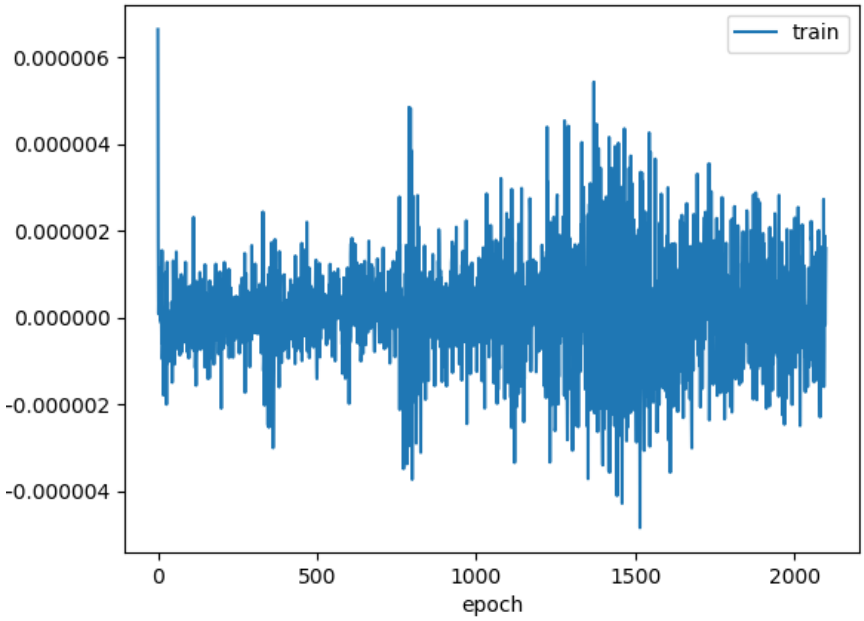

図 6.15　OX Discriminater 平均二乗誤差

第 7 章

関連研究

自然言語処理を数式を適応した例は筆者が調べるかぎりはなく，自然言語以外に Word2vec を適用しようとした研究はいくつかあり，再帰ニューラルネットを応用したものである．文献 [5] では文の特徴を Encoder-Decoder の機構を用いて学習し，Encoder の隠れ層の出力により文章の類似度を算出している．文献 [5] によると小説の一部を学習データに用いて，埋め込み層として SkipGram を適用している．gemsim の Doc2Vec との比較がなされているが，ユーザーにとって有益な評価方法の設定の難しさを述べている．

文献 [4] ではプログラムコードを埋め込み，再構成して実行できるかどうかを試している．C 言語で書かれたプログラムを一文字ごと，1-of-k 符号化して読み込んでいる．Dropout を適用させることにより性能を向上させている．今後，スペルミスによるコンパイルエラーを事前に直してくれることを目的としている．

文献 [6] では DeepLearing の技術の検索の際，ユーザーが求めている情報を推奨することを目的としている．評価方法として Movie Lens-100K での映画の推奨をパラメーター，ネットワークの構成を様々試しながら知見を集めている．AE は層を重ねても，性能は向上せず，各映画のユーザー平均を差し引いて正規化しても低脳は大きく変化しない．これにより，特徴抽出の重要さを筆者は述べている．

第 8 章

結論と今後の課題

本研究の結果によって系列変換により，人工的に作れた数式を意味のあるベクトルに変換する事ができた．これにより数式データのみで最適な問題選択を可能とし，いまある個人情報と合わせるとより高い精度でシステムが提供できるように思う．本研究により，自然言語処理の分野で発展してきた技術は人工言語でも利用することができることが示せた．これは人間が作り発展させてきたものはどこか本質的には同様の特徴を持っており，その特徴をコンピュータが学ぶことができたことではないだろうか．

しかしながら，未だ問題点はあり，計算問題には簡約な問題ほどパターンがなくなり，式から有用な情報が取り出せなくなる恐れがある．また，多くの子供が苦手とする文章題は本研究では扱っていないが，Dec2Vec など文章をベクトル化する手法は提案されており，それを応用することにより可能性は広がるだろう．また，本研究で扱った一次方程式の特徴量を使って別の分野を分類し，個人のまちがえた問題の情報を元に転移学習を行うとより，包括的なアダプティブラーニングシステムの構築が行えるようになり自分で学ぶ際の道しるべとなることを期待している．

謝辞

この研究を進めるにあたって，私の興味の赴くまま進める中，様々な角度からご指導いただいた千代教授に深く感謝申し上げます．また研究が，会話を通して気づきを与えてくれた同研究室メンバーに感謝申し上げます．最後に，毎週たくさんの計算問題を宿題として解いてきてくれた私の塾の生徒たちに感謝をするとともに必ず，この経験を元に世の中により良い教育システムを提供し今まで手の届かなかった子供達の未来に手助けすることを約束し，謝辞といたします．

参考文献

[1] Mikolov, T., Chen, K., Corrado, G. and Dean, J.: Efficient Estimation of Word Representations in Vector Space (2013).

[2] Sutskever, I., Vinyals, O. and Le, Q. V.: Sequence to Sequence Learning with Neural Networks (2014).

[3] ベネッセ教育総合研究所：第 6 回学習指導基本調査 DATA BOOK（高校版) (2016).

[4] 河原崎徳之岡田龍治：Deep Learning を用いた人工言語の学習と獲得について，*The 31st Annual Conference of the Japanese Society for Artificial Intelligence* (2017).

[5] 福田 清人 森 直樹 松本啓之亮：LSTM を用いた文の分散表現の獲得手法に関する一考察，言語処理学会 第 24 回年次大会発表論文集 (2018).

[6] 川上和也，松尾　豊：Deep Collaborative Filtering Deep Learning 技術推薦システムへ応用，*The 28th Annual Conference of the Japanese Society for Artificial Intelligence,* (2014).