卒業論文

計算問題の特徴分布にもと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

S152114 宮地 雄也

要旨

本研究は自然言語処理の技術を人工言語の数式に適用し，適切な数式の分類ができるかどうか調べることを目的としている．提案手法では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，さらに再帰ニューラルネットワークを用いて式のベクトルを得た．

提案手法を計算問題の復習を行う際の類題選出に利用し，生徒が間違えた問題の式に近い特徴をもつ式を選べることができるのを確認することができた．

現在主流なアダプティブラーニングの手法が大量の学習データを必要とするのに対して，本研究の成果を用いることで数式データのみで最適問題を選ぶことができ，復習に最適な類題選出を実現可能である．

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）（文献 [9]）の結果では平均正答率は小学校では算数 B が 51.7%，中学校では数学が 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．全国学力テストの中に小学校 6 年生時に算数が好きな生徒は 65.1% に対して 3 年後の中学校 3 年生の時でも数学が好きな割合が 51.6% と低く，勉強が進むにつれ苦手な子が増えることが分かる．

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ため，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してしまうことがあげられる．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いる．ベネッセ教育総合研究所の調査によると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 [7] 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割けていないことがわかる．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．

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ない．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おり一部の大手企業が情報を独占している．

そこで個人のデータではなく，解く数式の方に着目し，生徒が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し，そのベクトルを用いて実際に復習問題選出を行う手法を提案する．

数式自体も自然言語と同様にある一定のルールがあり，自然言語処理を使ってベクトル化する手法は自然な発想である．さらにこの手法を用いれば現在主流の大量の個人データを必要とせず，問題データさえあれば良いので，比較的に簡単に利用することができる利点がある．

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

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 を，i 番目の要素のみ 1 で残りが全て 0 になっている次元 V のベクトルで表したものである（式 2.1）．

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

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る．一つ目は，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である．二つ目は 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点である．疎なベクトルとは次元数が大きくても 0 以外の要素がほとんどないベクトルをさす．

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 ベクトルから密なベクトルを作り出し，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，ベクトルの次元数を削減することができる．

2.1 節，2.2 節に本研究で用いる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．これら以外に文献 [2] で紹介されている fasttext と呼ばれる手法や，共起行列と組み合わせた Glove（文献 [1]）などがある．

2.1 Continuous Bag-of-Words Model

Continuous Bag-of-Words Model（以降，CBOW と略記）は文献 [4] で提唱された手法である．ある単語数 の単語列 $w_1, w_2, w_3, \ldots, w_{T-2}, w_{T-1}, w_T$ 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 において w_t をターゲットにした時の前後単語 m 個をコンテキストとする事後確率は式 2.2 のように書くことができる．つまり 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 にコーパス全体に拡張した形を示す．

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

$$L = -\frac{1}{T}\sum_{t=1}^{T} \log P(w_t | w_{t-m}, ..., w_{t+m}) \quad T \text{ は単語数を表す} \tag{2.3}$$

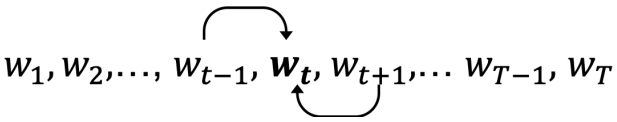

図 2.1　単語の列からターゲットとなる単語を推測する ($m = 1$)

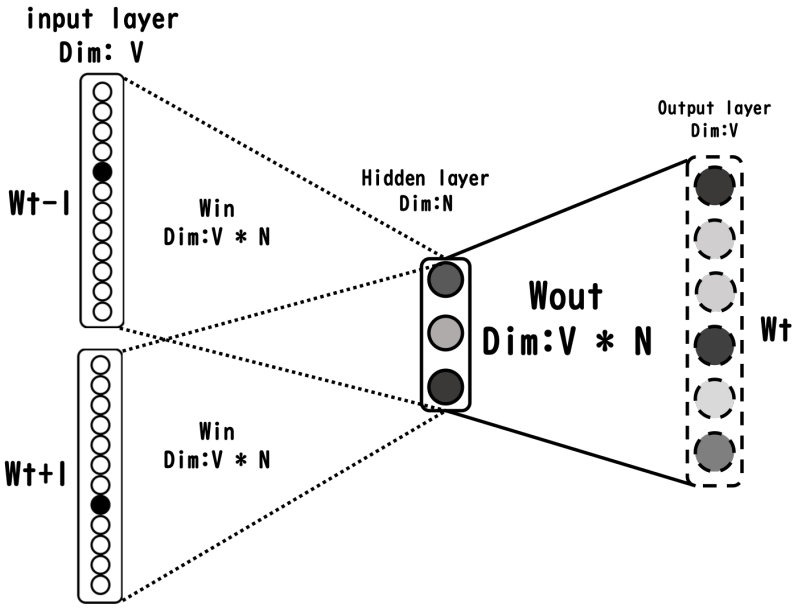

図 2.2　CBOW のネットワーク構成モデル模式図 ($m = 1$)

2.2　Skip Gram

Skip Gram は文献 [4] で紹介されている 2.1 章で述べた CBOW とは別の手法である．CBOW とは逆に中心の単語 w_t から前後 m 個の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 ($m = 1$)

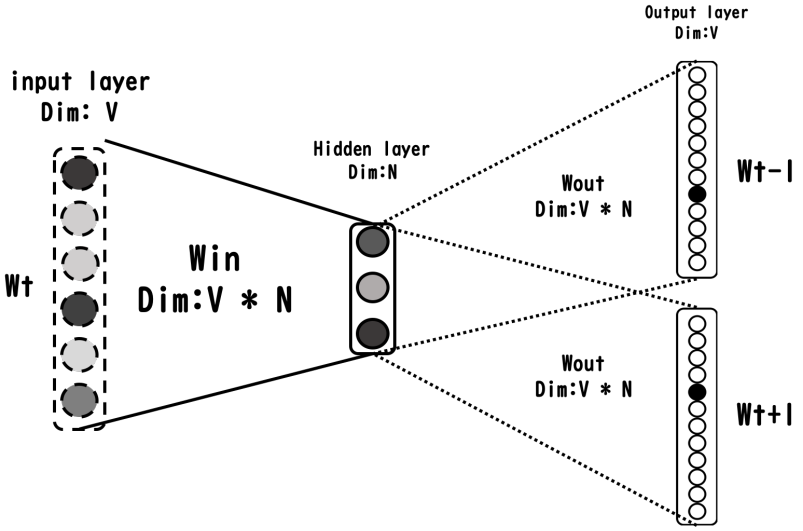

図 2.4　SkipGram のネットワーク構成モデル模式図 ($m = 1$)

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．

$$\begin{aligned} -\log P(w_{t-m}, ..., w_{t+m}|w_t) &= -\log \prod_{k=1}^{m} P(w_{t-k}|w_t) \log P(w_{t+k}|w_t) \\ &= -\sum_{k=1}^{m} (\log P(w_{t-k}|w_t) + \log P(w_{t+k}|w_t)) \end{aligned} \tag{2.5}$$

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．

$$\begin{aligned} -\sum_{C} \log P(w_{t-m}, ..., w_{t+m}|w_t) &= \sum_{C} -\log \prod_{k=1}^{m} P(w_{t-k}|w_t) P(w_{t+k}|w_t) \\ &= -\sum_{C} \sum_{k=1}^{m} \log P(w_{t-k}|w_t) + \log P(w_{t+k}|w_t) \end{aligned} \tag{2.6}$$

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．

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}，現時刻の x_t を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．その重みの計算には入力と一時刻前の隠れ層の出力が使われている（式 3.1）．

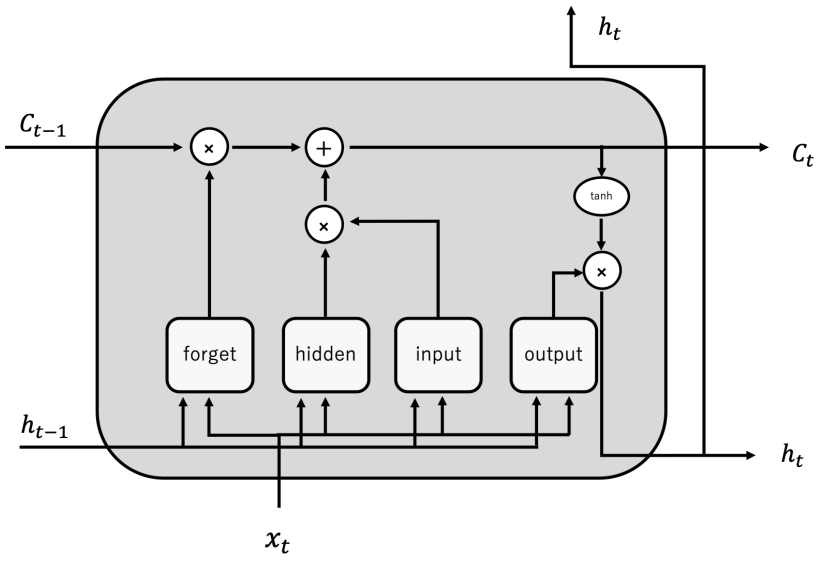

図 3.1　LSTM のネットワーク構成モデル模式図：図中の forget,hidden,input output はそれぞれゲートが扱う情報を示している

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

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．その重みの計算には入力と一時刻前の隠れ層の出力が使われている（式 3.2）．

$$o = sigmoid(x_t W_x^{(o)} + h_{t-1} W_h^{(o)} + b^{(o)}) \tag{3.2}$$

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 3.3）．

$$\bar{h} = \tanh(x_t W_x^{(h)} + h_{t-1} W_h^{(h)} + b^{(h)}) \tag{3.3}$$

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．その重みの計算には入力と一時刻前の隠れ層の出力が使われている（式 3.4）．

$$i = sigmoid(x_t W_x^{(i)} + h_{t-1} W_h^{(i)} + b^{(i)}) \tag{3.4}$$

3.5 LSTM のまとめ

3.1 節，3.2 節，3.4 節，3.3 節で紹介したものをまとめると式 3.5 のようになる．

$$\begin{aligned} f &= sigmoid(x_t W_x^{(f)} + h_{t-1} W_h^{(f)} + b^{(f)}) \\ o &= sigmoid(x_t W_x^{(o)} + h_{t-1} W_h^{(o)} + b^{(o)}) \\ \bar{h} &= \tanh(x_t W_x^{(h)} + h_{t-1} W_h^{(h)} + b^{(h)}) \\ i &= sigmoid(x_t W_x^{(i)} + h_{t-1} W_h^{(i)} + b^{(i)}) \end{aligned} \tag{3.5}$$

$$C_t = f \odot c_{t-1} + g \odot i \tag{3.6}$$

$$h_t = o \odot \tanh(C_t) \tag{3.7}$$

LSTM モデルを 3 時刻分展開したものを図 3.2 に示す．

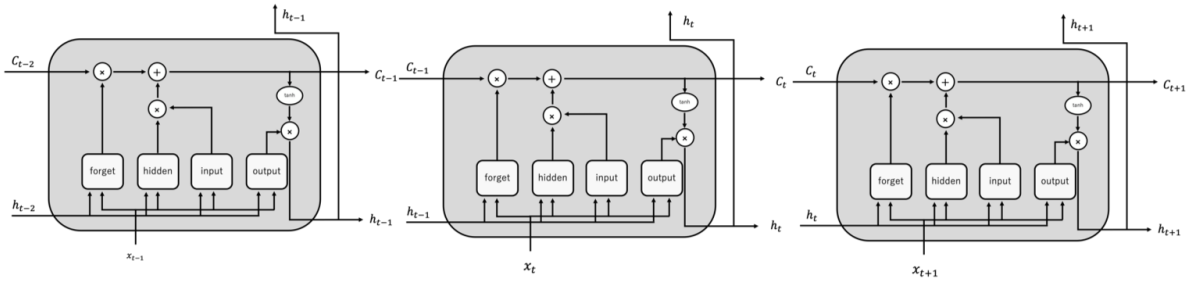

図 3.2　LSTM の 3 時刻展開:記憶を守る機構によりより高い濃度で次の時刻に情報を渡すことができる

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．

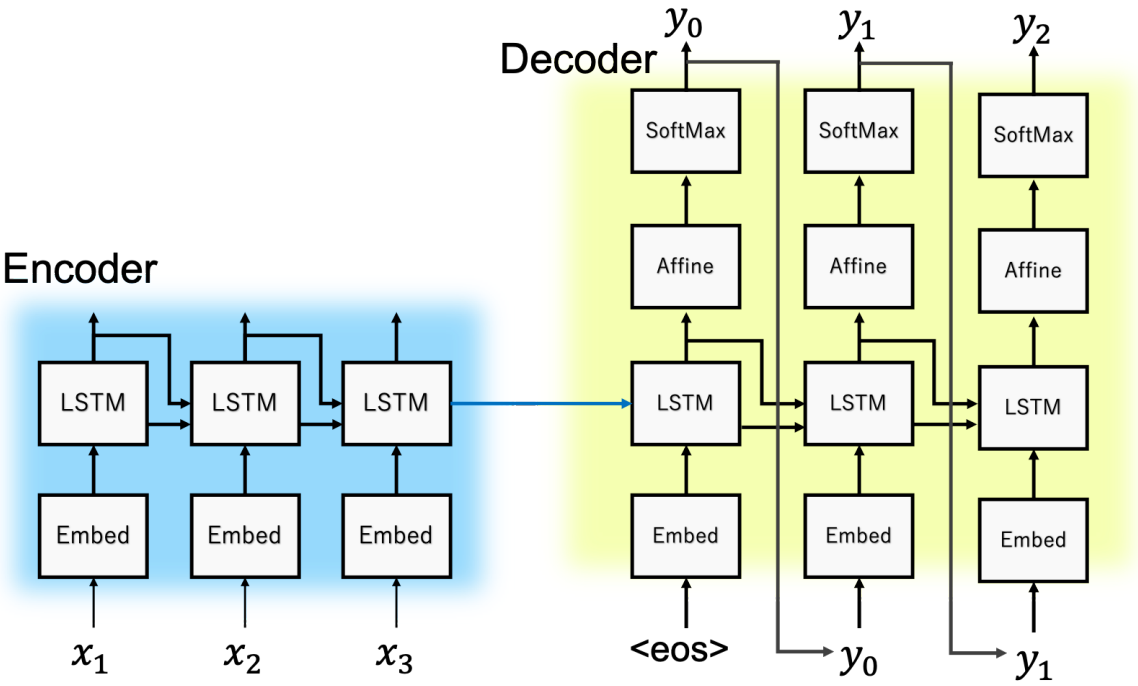

図 4.1　LSTM を用いた基本的な系列変換モデルの 3 時刻展開

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，区切り文字 <EOS> を入力として受け取り，Encoder 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．

$$softmax(x)_i = \frac{e^{x_i}}{\sum_i e^{x_i}} \tag{4.1}$$

区切り文字 <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 を出力し再び区切り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を扱う．

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の概要を図 5.1 に示す，ExericisesEncoder は数式を数式をベクトル 表現に変換し蓄積する．その変換結果のベクトルと間違えた問題も同様の手段でベクトル化し蓄積した式のベクトル表現との類似度を算出し，その類似度が高い式を類題として出力する．

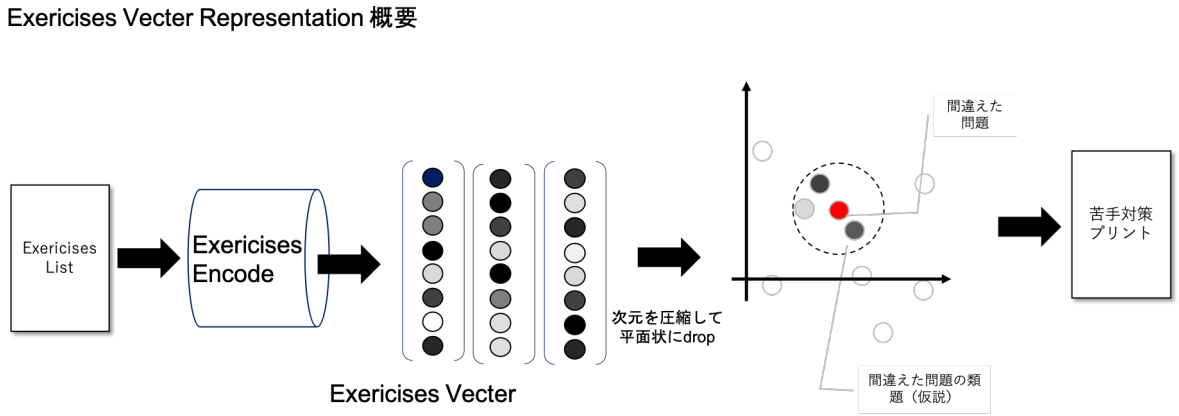

図 5.1　ExericisesEncoder　模式図

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，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，文字，小数点，符号，丸括弧，波括弧でわけそれぞれ onehot 表記でベクトル化する．

実験で用いた分散表現獲得手法は以下の 2 手法である．

- CBOW
- SkipGram

5.2.2　文字分布の予備実験結果

予備実験として埋め込み層に利用する CBOW，SkipGram で文字の特徴が得られているかを確認した．学習データとして 350 種類の一次方程式の問題を用いた．その一部を以下に示す．

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

以降，\frac を f と表記し，上記の式を $f, \{, \}, (,), x, =, +, -, ., 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,$ <EOS>,<UNK> を語彙として onehot ベクトルに変換した．埋め込みベクトルの次元数は 200 次元とした．結果を図 5.2〜 図 5.5 に示す

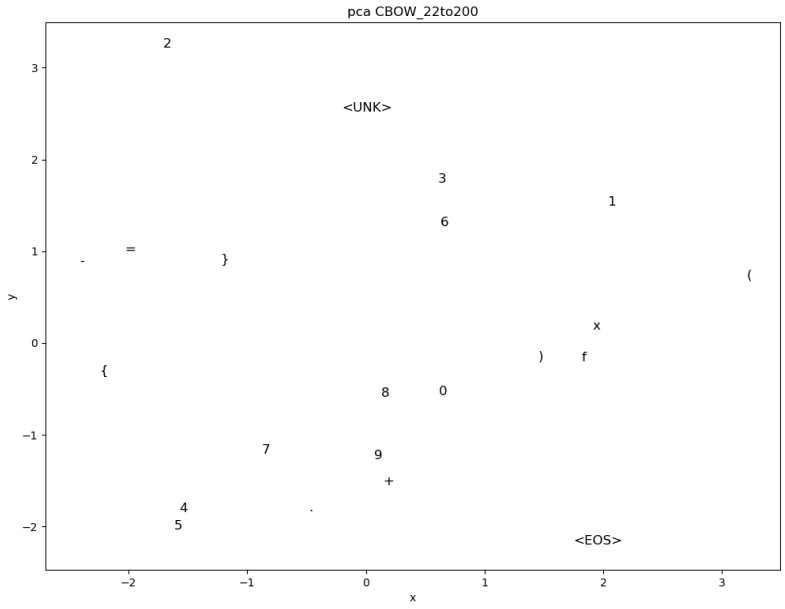

図 5.2　CBOW にて 22 次元を 200 次元に分散表現を変換したのち PCA で再構成しグラフ化した結果

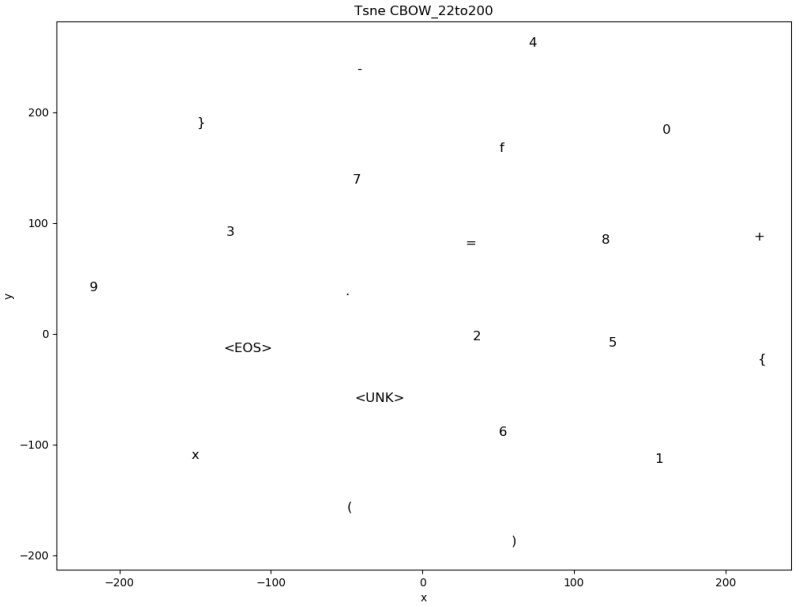

図 5.3　CBOW 22 次元を 200 次元に分散表現を変換したのち t-SNE で再構成しグラフ化した結果

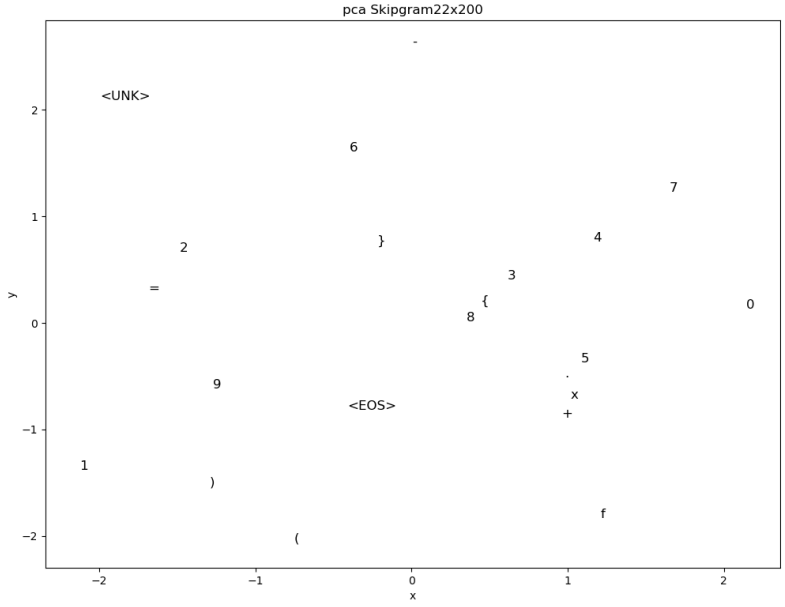

図 5.4　SkipGram にて 22 次元を 200 次元に分散表現を変換したのち PCA で再構成しグラフ化した結果

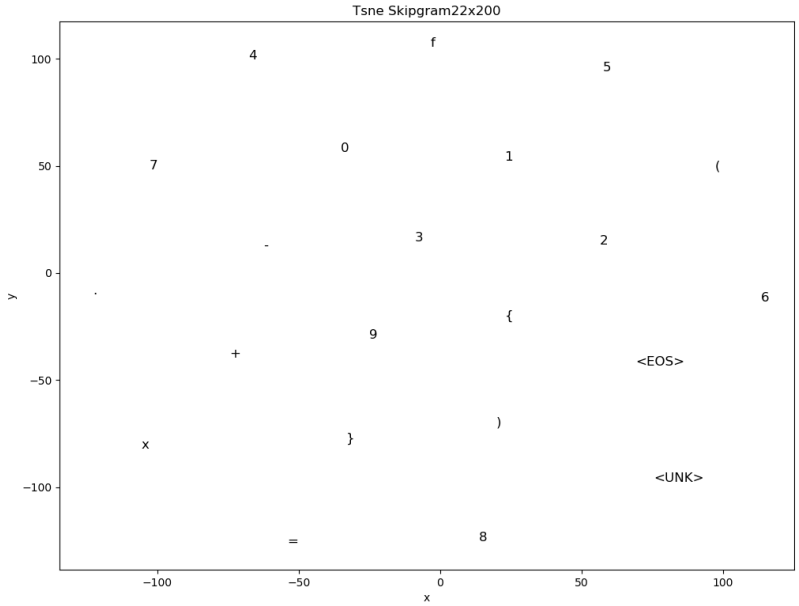

図 5.5　SkipGram にて 22 次元を 200 次元に分散表現を変換したのち t-SNE で再構成しグラフ化した結果

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 と書かれている横の文字が入力文字でその下に類似度の高い順に 5 つ並んでいる．例えば f を入力した時では最も類似度が高いのが同じ文字の f であり，その下に 1，(，0，9 と続く．

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

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

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

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

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

input -> x
x > : 1.0
3 > : 0.10958019
```

```
1 > : 0.09617383
<EOS> > : 0.06585052
4 > : 0.04960839
```

```
3 > : 0.107833974
x > : 0.09617383
) > : 0.092290804
```

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

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

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

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

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

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

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

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

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

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

```
input -> 1
1 > : 1.0
f > : 0.17398714
```

```
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
```

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．文献 [5] の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

$$(\quad と \quad) \text{の距離 } 27.04968437532474$$

$$\{\quad と \quad\} \text{の距離 } 27.04968437532474$$

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，　数字自体の演算はできないが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 を用いて求めた．

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

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

5.3　系列変換モデルの検討

本論文では系列変換の過程で用いる 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．

- 通常の LSTM で行う系列変換を多層に積み上げるモデル（図 5.6）
- 通常 Decoder が側で利用される SkipConnection を Encoder で採用した SkipConnection モデル（図 5.7）
- 前時刻の情報も用いる Bi-Directional（図 5.8）．

この三つを用いて実験を行った．

今回，この実験では Encoder 側がより精度の高いベクトルを生成してもらうことが目的なので Decoder 側は通常の LSTM を一層で行う系列変換で確認を行うこととした．

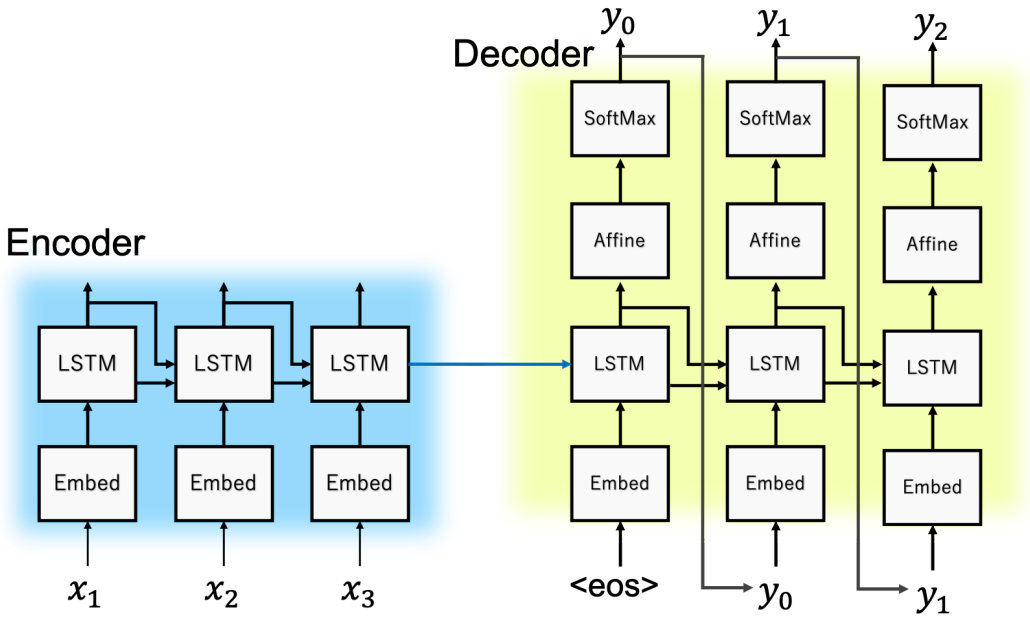

図 5.6　基本モデル

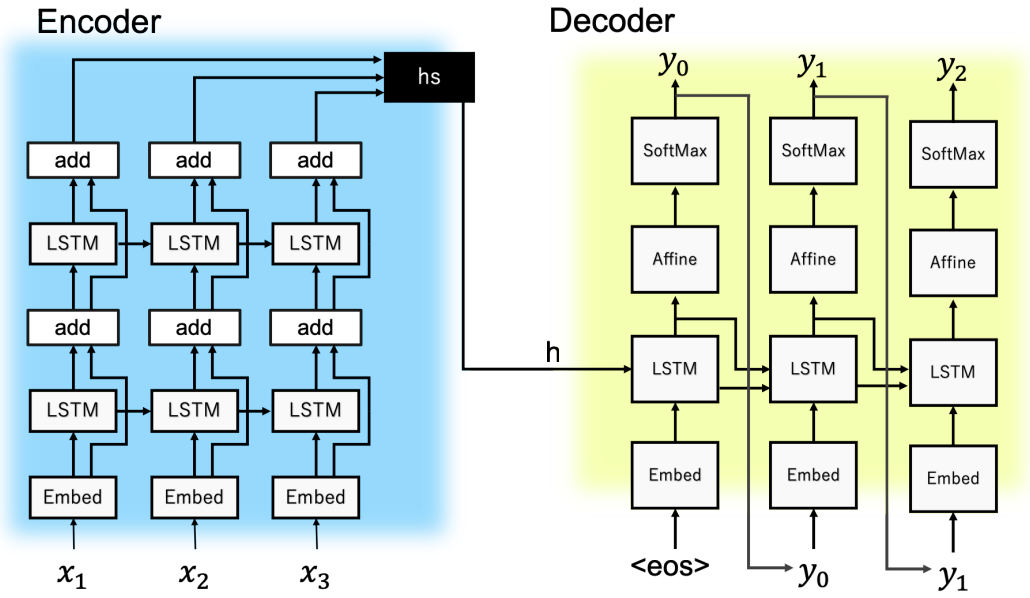

図 5.7　SkipConnection モデル

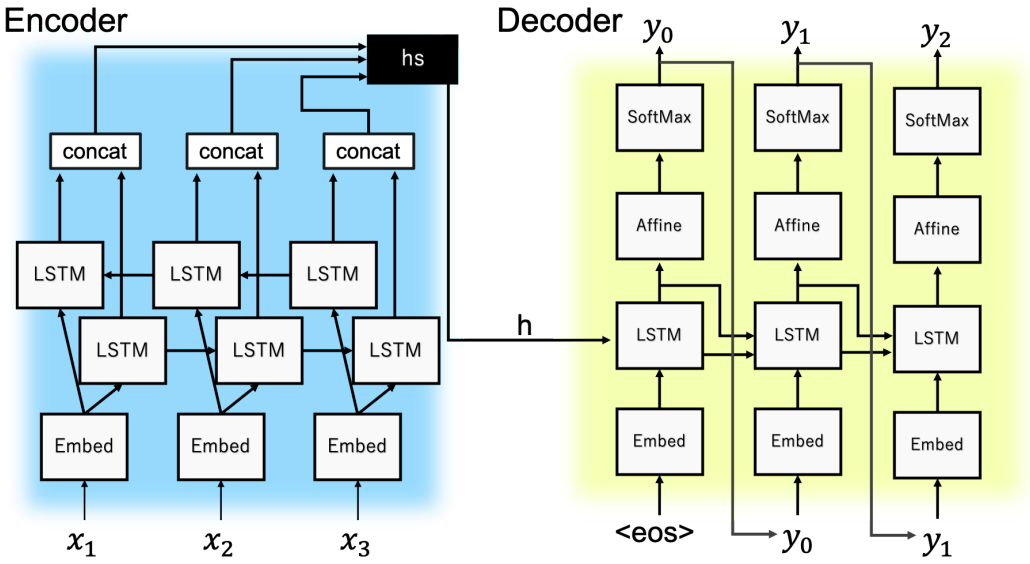

図 5.8　Bi-Directionl モデル

第 6 章

結果とその検討

6.1　実験内容

以降では今回の実験内容（目的および方法）について説明する．

6.1.1　実験 1 ：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 の出力値を PCA，t-SNE で二次元ベクトルに圧縮し図示し確認する

実験で試したネットワーク構成を表 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

6.1.2 実験２：計算式の類題選出

- 目的．数式のベクトル表現は類題選出として利用可能かどうかの検証
- 方法．以下の 5 種類の式に近い式を算出した．式の選定基準は一つ目は単純な式，二つ目は分数を含む式，三つ目は小数を含む式，四つ目はかっこが含まれる式，五つ目は四つ目のかっこの前の係数だけ変更した式となっている．これらにより類題として選出されるものがどう変化するかを確認した．100 種類の方程式を表 6.1 で示したネットワークでベクトル化を行なったのち，学習した式との類似土を計算する．算出には cos 類似度を用いて求める．
  1. `x-3=7`
  2. `f{4}{3}x-9=-10`
  3. `0.1x=0.5x-1.6`
  4. `2(x+5)=6`
  5. `-4(x+5)=6`

6.2 実験結果

6.2.1 実験１：計算式の特徴量抽出

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 20 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`
- `-\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`

この 20 個の式を学習させ，PCA を用いて 2 次元に圧縮し二次元座標平面状にプロットしたものが以下の図 6.1 から図 6.10 である．

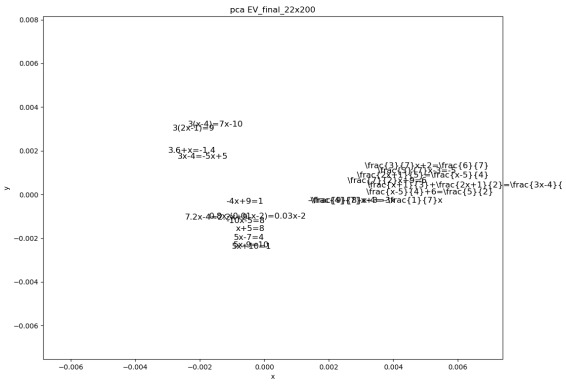

図 6.1　Normal　LSTM 1 層 200 次元　学習データ

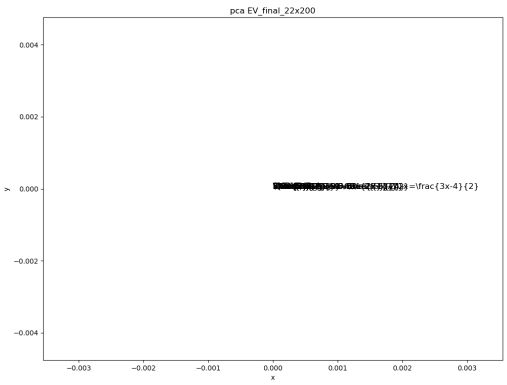

図 6.2　Normal　LSTM4 層 200 次元　学習データ

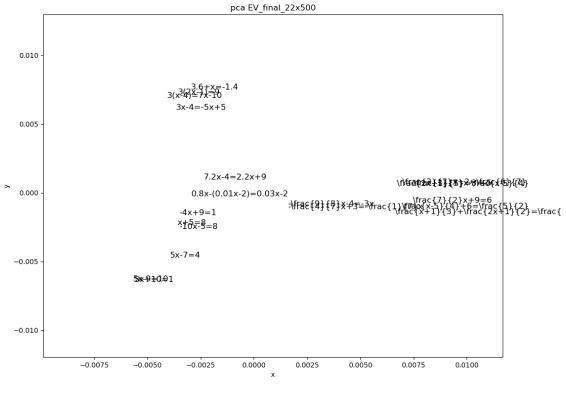

図 6.3　Normal　LSTM 1 層 500 次元　学習データ

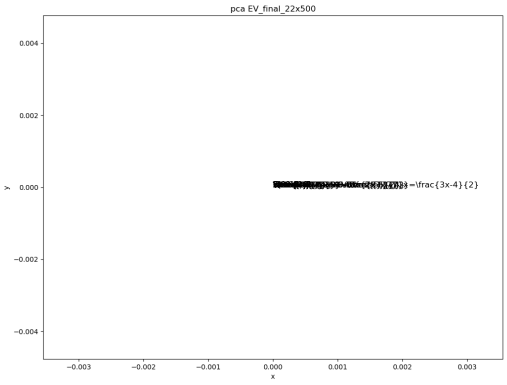

図 6.4　Normal　LSTM4 層 500 次元　学習データ

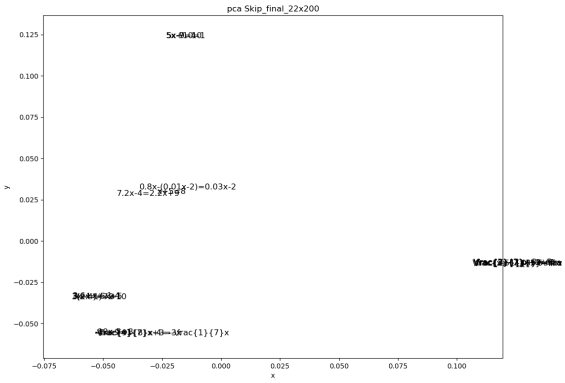

図 6.5　Skipconnect : LSTM1 層 200 次元　学習データ

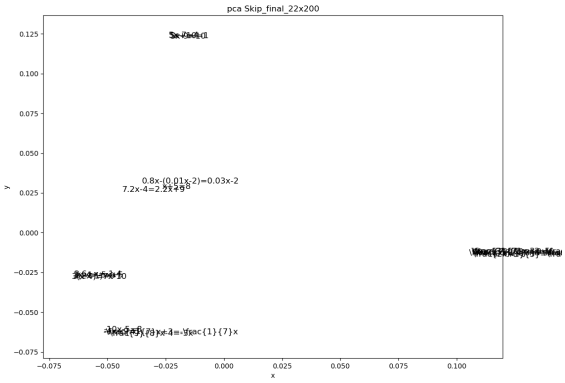

図 6.6　Skipconnect : LSTM 4 層 200 次元　学習データ

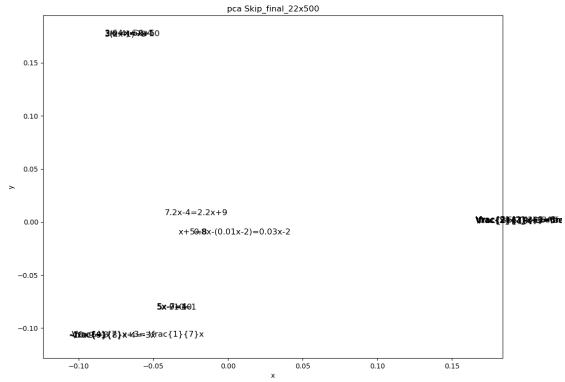

図 6.7　Skipconnect : LSTM1 層 500 次元　学習データ

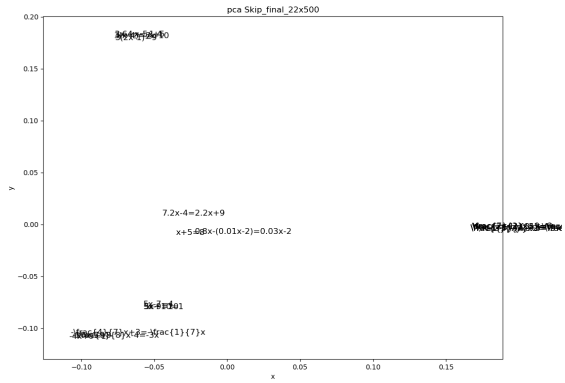

図 6.8　Normal　LSTM4 層 500 次元　学習データ

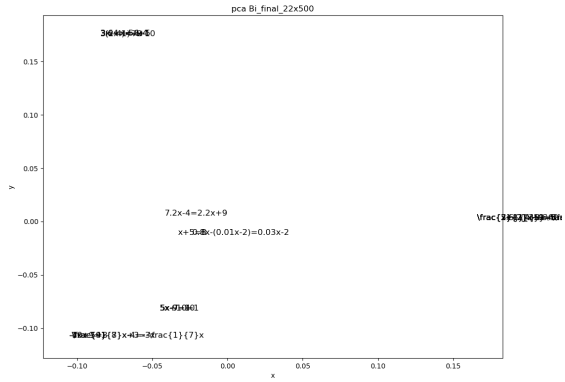

図 6.9　BiDirection : LSTM1 層 500 次元　学習データ

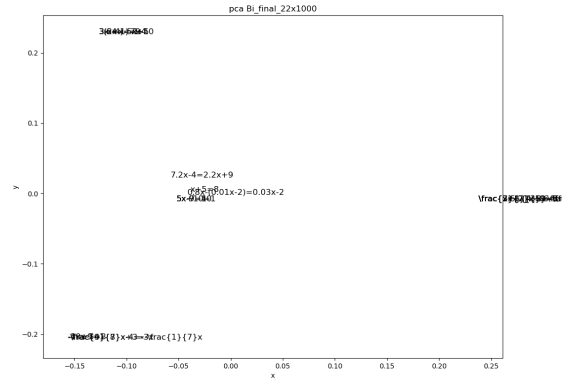

図 6.10　BiDirection : LSTM1 層 1000 次元　学習データ

6.2.2 実験２：計算式の類題選出

表 6.1 で示したネットワークごとに並べている．冒頭に（学習ネットワーク方法，次元数，LSTM 層の層数）で学習方法を示している，全ての結果は付録 A に掲載する．各モデルの $-4(x+5)=6$ の実行結果である．

```
mode: E2D size: 22x200 layer: 1
input -> -4(x+5)=6
-4x+5=-11 > 0.956455
-3(4-x)=x > 0.8782562
-10x-9=-5 > 0.76801944
-10x+6=-5 > 0.7587959
-2x-3=6x+13 > 0.6597776

mode: E2D size: 22x200 layer: 4
input -> -4(x+5)=6
-3(4-x)=x > 0.7466616
-4x+5=-11 > 0.71639407
4x+7=x-11 > 0.61074364
5x=2(x-9) > 0.59230787
-2(x+8)=6x-6 > 0.5850883

mode: E2D size: 22x200 layer: 8
input -> -4(x+5)=6
9x-8=5x+6 > 0.50285524
-10x+6=-5 > 0.45350274
-3(4-x)=x > 0.4236934
0.2(x+6)=0.5x-0.6 > 0.39269254
f{1}{3}x+0.5x=f{10}{9} > 0.38043457

mode: E2D size: 22x500 layer: 1
input -> -4(x+5)=6
-4x+5=-11 > 0.95100325
-3(4-x)=x > 0.8223118
-10x-9=-5 > 0.73558027
-10x+6=-5 > 0.7320987
-4x+9=1 > 0.60368705
```

```
mode: E2D size: 22x500 layer: 4
input -> -4(x+5)=6
2x+5=8x+4 > 0.6925208
-4x+5=-11 > 0.67323875
3x+5=-x+2 > 0.6688769
-3(4-x)=x > 0.6653332
-10x+6=-5 > 0.59487367


mode: E2D size: 22x500 layer: 8
input -> -4(x+5)=6
9x-8=5x+6 > 0.7264374
-10x+6=-5 > 0.69968903
-3(4-x)=x > 0.5661228
2x+5=8x+4 > 0.5191195
-2(x+6)=5(x-4) > 0.4562482

mode: E2D size: 22x1000 layer: 1
input -> -4(x+5)=6
-4x+5=-11 > 0.9619249
-3(4-x)=x > 0.8356369
-10x+6=-5 > 0.75671685
-10x-9=-5 > 0.7534554
-4x+9=1 > 0.63603586

mode: E2D size: 22x1000 layer: 4
input -> -4(x+5)=6
-4x+5=-11 > 0.6981378
-3(4-x)=x > 0.6155417
3x+5=-x+2 > 0.55136484
-10x+6=-5 > 0.5324473
2x+5=8x+4 > 0.5307147

mode: E2D size: 22x1000 layer: 8
input -> -4(x+5)=6
9x-8=5x+6 > 0.6988591
-10x+6=-5 > 0.6161238
-3(4-x)=x > 0.47893324
2(x+5)=6 > 0.46710303
2x+5=8x+4 > 0.4185045
```

```
mode: SkipConnect size: 22x200 layer: 1
input -> -4(x+5)=6
-4x+5=-11 > 0.9999555
-3(4-x)=x > 0.99987245
-10x+6=-5 > 0.99976254
-10x-9=-5 > 0.99975955
-5(x-1)=-7(x+3) > 0.9995558

mode: SkipConnect size: 22x200 layer: 4
input -> -4(x+5)=6
-4x+5=-11 > 0.99926853
-3(4-x)=x > 0.99795955
-10x+6=-5 > 0.9962721
-10x-9=-5 > 0.99624175
-5(x-1)=-7(x+3) > 0.9928169

mode: SkipConnect size: 22x200 layer: 8
input -> -4(x+5)=6
-4x+5=-11 > 0.9969909
-3(4-x)=x > 0.9919083
-10x+6=-5 > 0.9856282
-10x-9=-5 > 0.98561376
-5x=20 > 0.97107166

mode: SkipConnect size: 22x500 layer: 1
input -> -4(x+5)=6
-4x+5=-11 > 0.9998899
-3(4-x)=x > 0.9996674
-10x+6=-5 > 0.9994921
-10x-9=-5 > 0.99945056
-4x+9=1 > 0.9989111

mode: SkipConnect size: 22x500 layer: 4
input -> -4(x+5)=6
-4x+5=-11 > 0.99834
-3(4-x)=x > 0.9950138
-10x+6=-5 > 0.99222285
-10x-9=-5 > 0.99158734
-4x+9=1 > 0.98335236
```

```
mode: SkipConnect size: 22x500 layer: 8
input -> -4(x+5)=6
-4x+5=-11 > 0.99358886
-3(4-x)=x > 0.98175734
-10x+6=-5 > 0.97098434
-10x-9=-5 > 0.9683183
-4x+9=1 > 0.93842256

mode: SkipConnect size: 22x1000 layer: 1
input -> -4(x+5)=6
-4x+5=-11 > 0.9994779
-3(4-x)=x > 0.9991171
-10x+6=-5 > 0.9987582
-10x-9=-5 > 0.9984923
-4x+9=1 > 0.99770147

mode: SkipConnect size: 22x1000 layer: 4
input -> -4(x+5)=6
-4x+5=-11 > 0.99144673
-3(4-x)=x > 0.9864846
-10x+6=-5 > 0.98037875
-10x-9=-5 > 0.9759923
-4x+9=1 > 0.9651148

mode: SkipConnect size: 22x1000 layer: 8
input -> -4(x+5)=6
-4x+5=-11 > 0.95918113
-3(4-x)=x > 0.949258
-10x+6=-5 > 0.9239827
-10x-9=-5 > 0.90235525
-4x+9=1 > 0.8695013

mode: BiDirection size: 22x200 layer: 1
input -> -4(x+5)=6
-2(x+6)=5(x-4) > 0.99999905
-f{1}{3}(9x+24)+6(f{1}{3}x-1)=f{1}{4}(16x+8)+3x-1 > 0.9999986
-3(2x-8)=4(x+1) > 0.99999833
-10x+6=-5 > 0.9999983
-5(x-1)=-7(x+3) > 0.9999983
```

```
mode: BiDirection size: 22x500 layer: 1
input -> -4(x+5)=6
-2(x+6)=5(x-4) > 0.9999838
-3(2x-8)=4(x+1) > 0.9999718
-2(x-5)=9x-8 > 0.99997026
-2(x+8)=6x-6 > 0.99996954
-10x+6=-5 > 0.9999664

mode: BiDirection size: 22x1000 layer: 1
input -> -4(x+5)=6
-2(x+6)=5(x-4) > 0.999892
-3(2x-8)=4(x+1) > 0.99982303
-f{1}{3}(9x+24)+6(f{1}{3}x-1)=f{1}{4}(16x+8)+3x-1 > 0.99979335
-2(x+8)=6x-6 > 0.99978095
-0.4(f{5}{2}x+12)=-3 > 0.9997515
```

6.3 考察

6.3.1 実験１：計算式の特徴量抽出について

今回の実験では 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 を深くすると勾配が消失し，一点に集中している（図 6.2，図 6.4）．しかし，図 6.2 と図 6.6，図 6.4 と図 6.8 を比較してみると SkipConnection を用いることで多層 LSTM で学習しても過学習が発生することを防ぐ事が確認できた．

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．

図 6.1，図 6.2，図 6.3，図 6.4，図 6.5，図 6.6，図 6.7，図 6.8，図 6.9, 図 6.10, のどれもが同系列そうな問題が近い分布となった．図 6.1 は分数の中でも符号でも分かれており，拡大してみてみると式の始まりが負の方程式だけまとまっているところがありわずかながらその違いを表しているように見える（図 6.11）.

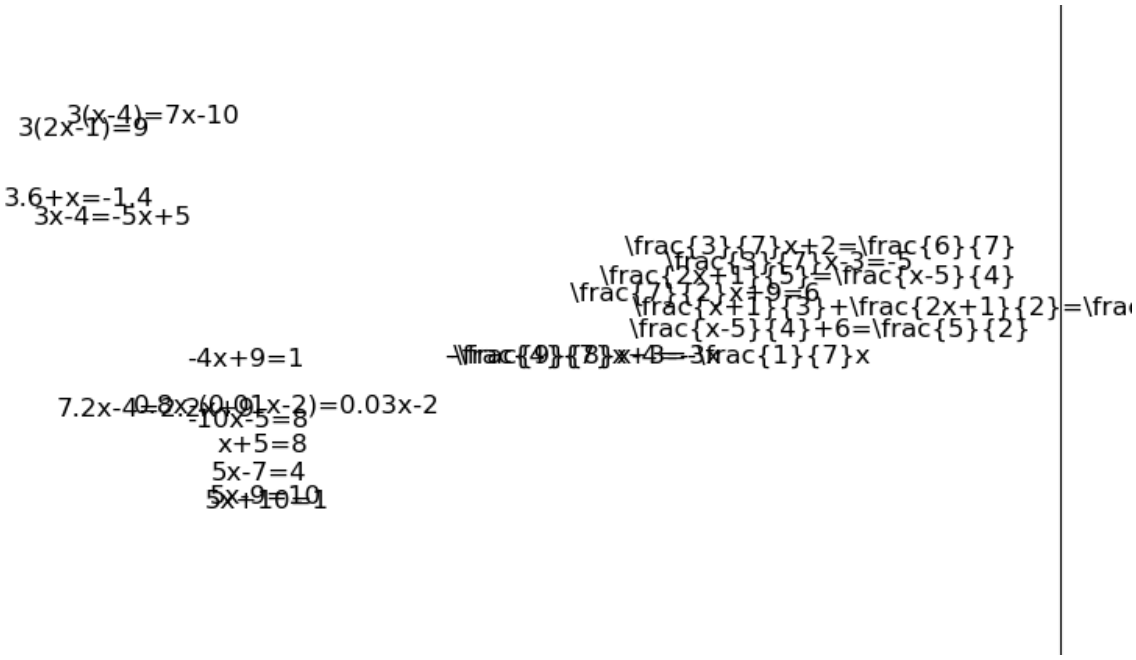

図 6.11　NormalRNN で作った 200 次元の ExericisesVecter の一部

特に図 6.12 と図 6.13 をみると Bi-DirectionalRNN で作った 500 次元の ExericisesVecter の方が小数を含む式と整数だけの式の距離がより離れ，分類がより正確になっているように見える．このことからより高次元で再構成すると高い再現率が得られるとは言えず．繊細なチューニングが不可欠である．

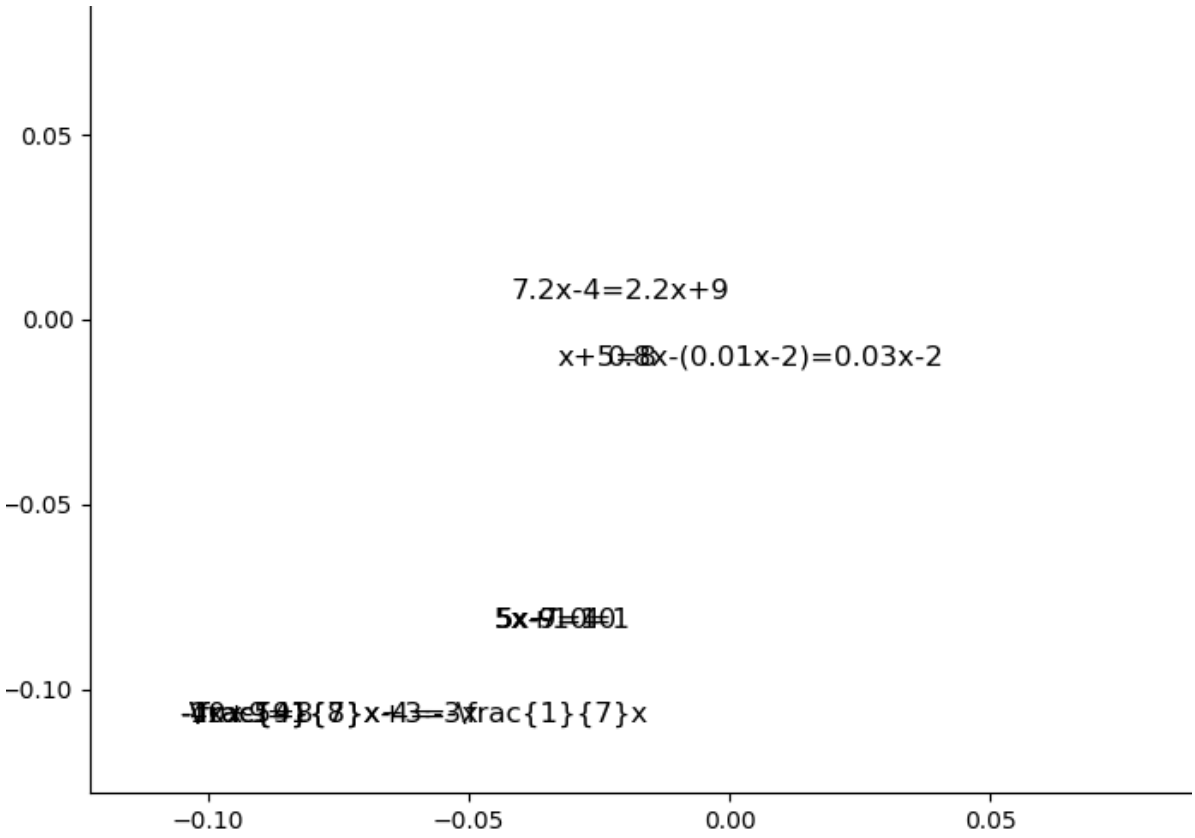

図 6.12　Bi-DirectionalRNN で作った 500 次元の ExericisesVecter の一部

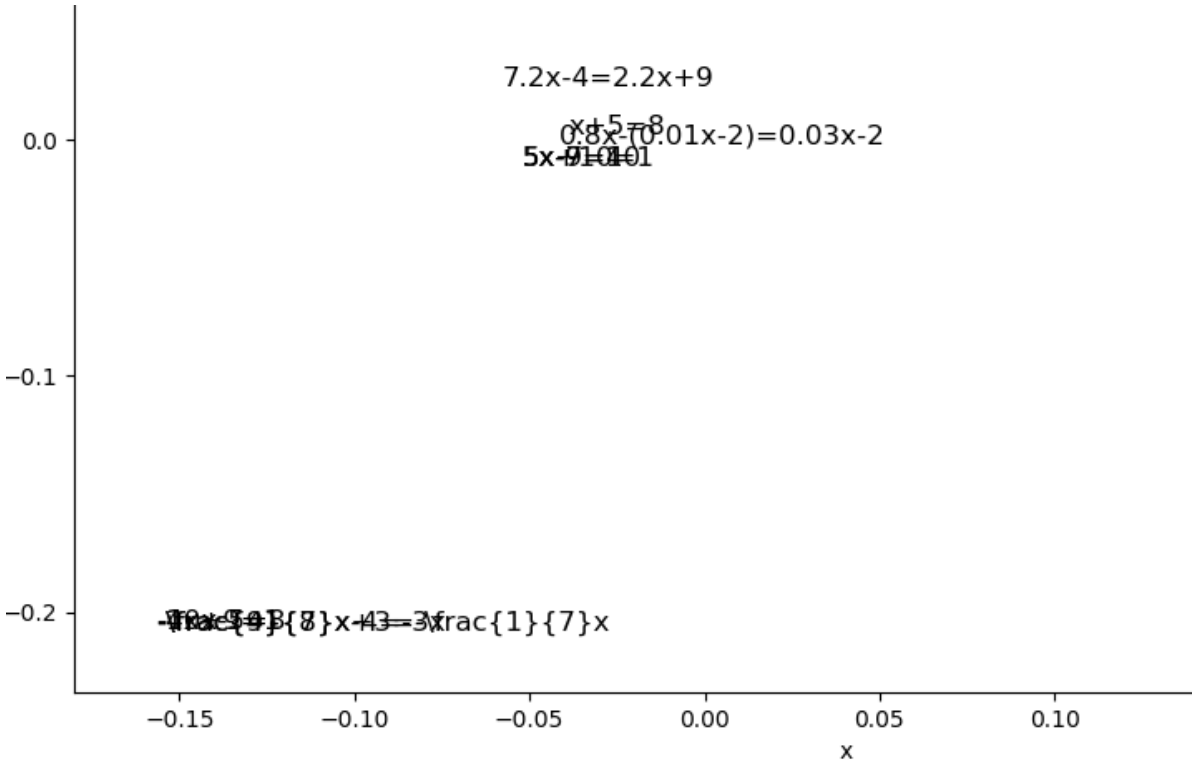

図 6.13　Bi-DirectionalRNN で作った 1000 次元の ExericisesVecter の一部

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，SkipConnect では層を深くしても過学習は起こらないが特徴をうまく捉え切ることができなかった．

6.3.2 実験２：計算式の類題選出

どのモデルパターンも単純な $x-3=7$ は同じような式が類似度トップ 5 に並んでいる．各モデルパターンでは次元数の 200，500，1000 では類似度に違いはあっても選出される式に違いはなかった．

$-4(x+5)=6$ はネットワークごとに違いが出ており normal の 500 次元 1 層や SkipConnect の 500 次元 4 層ではカッコを使った式の出現は少ないものの Bi-Directional200 次元一層ではカッコがあるものが 8 割を占めており，何らかの特徴を抽出できているのではないかと考える．この結果から式の特徴抽出には双方向 RNN が有効であることがわかる．

第 7 章

関連研究

自然言語処理を数式を適用した例は筆者が調べるかぎりは存在しない．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る．文献 [12] では文の特徴を Encoder-Decoder の機構を用いて学習し，本研究と同様に 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 [12] 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．gensim の Doc2Vec との比較がなされているが，ユーザにとって有益な評価方法の設定の難しさを述べている．これらの文章をベクトル化するというのは文献 [3] から始まり注目を集めている．

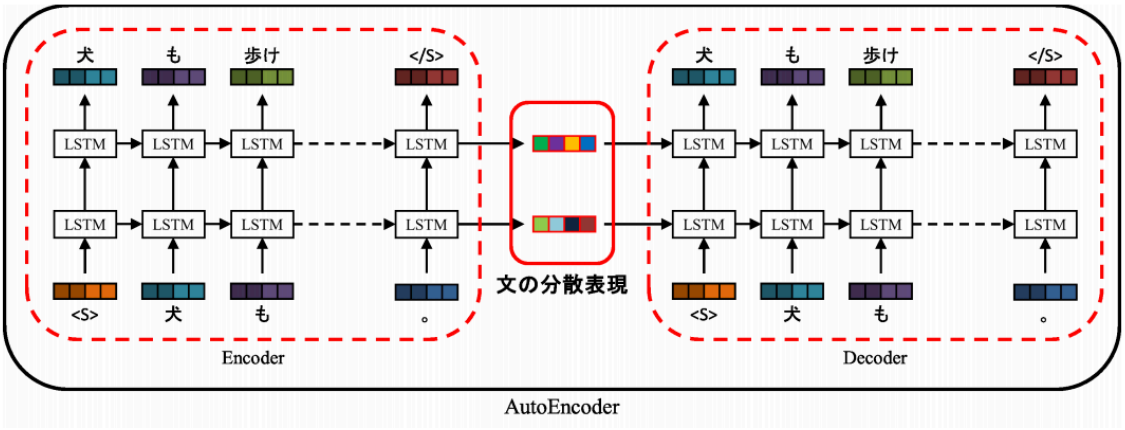

図 7.1　文献 [12]　ネットワーク構成

文献 [10] ではエラーのあるプログラムコードを埋め込み，再構成して実行できるプログラムを再構成できるか試している．C 言語で書かれたプログラムを一文字ごと，onehot 化して読み込んでいる．この論文では大小文字を区別して学習する．1-D Convolution を用いて情報を圧縮し再構成することで上記のことが可能だと述べている．Dropout を用いると性能が向上しており，自然言語と同様に過学習が通常の RNN では起きるので，対策として本論文でも Dropout を採用した．

文献 [8] では DeepLearing の技術の検索の際，ユーザ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，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．AutoEncoder は層を重ねても，性能は向上せず，各映画のユーザ平均を差し引いて正規化しても性能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．

文献 [6] は漢字などの象形文字から文字の分散表現を得る研究である．図 7.2 で示すように，画

像データを畳み込みニューラルネットを持ちいて次元圧縮していき，最後には 1024 次元のベクトルとして出力する．この最終層の出力を漢字の分散表現としてもちいて他の自然言語タスクを従来方法より超える結果を出している．特徴として漢字を現在使われている字体だけでなく，旧字体も用いて性能向上を図っている（図 7.3）．

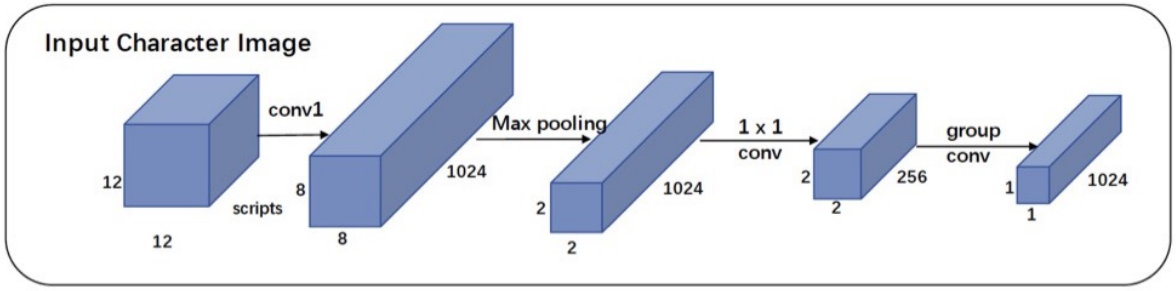

図 7.2　文献 [6]　ネットワーク構成

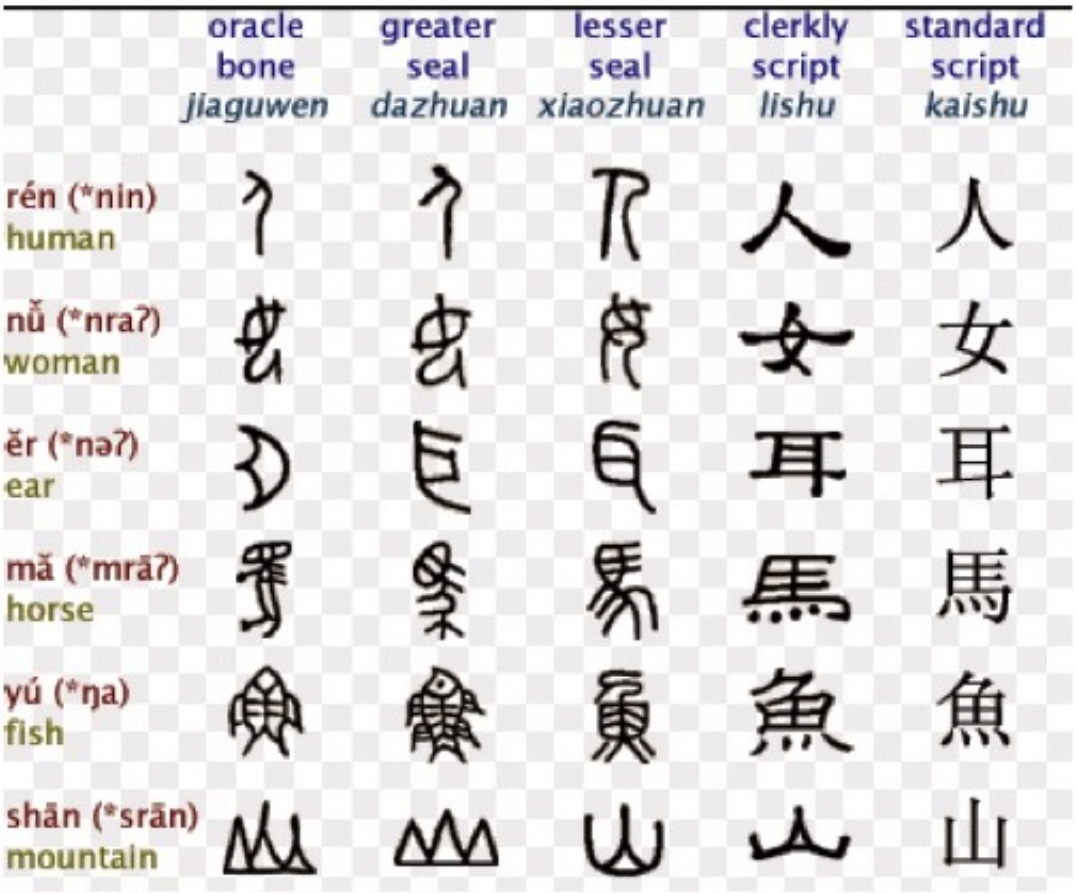

図 7.3　象形文字も学習に使い，その文字がどのように変化してきたのかも考慮する

第 8 章

結論と今後の課題

本研究では自然言語処理の技術を人工言語の数式に適用し，分類ができるかどうかを目的とした．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を行い，式のベクトルを得た．この結果から現在では間違った問題から人をグ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．

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．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るということではないだろうか．

しかしながら，未だ問題点はある．計算問題には優しい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．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，Doc2Vec 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．本研究で扱った一次方程式の特徴量を使って別の分野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ることを期待している．

付録 A

実験２：計算式の類題選出の実行結果

```
mode: E2D size: 22x200 layer: 1
input -> x-3=7
x-4=7 > 0.95820755
x-5=5 > 0.9391084
x+2=2 > 0.7650795
x+5=8 > 0.7316054
x-8=-3 > 0.70472425

input -> f{4}{3}x-9=-10
f{4}{3}x-9=-10 > 0.99999994
f{3}{2}x-6=-5x > 0.9709359
f{7}{3}-7x=-3x > 0.9628508
f{10}{9}x-3=-5 > 0.9467073
f{4}{3}x+7=-2 > 0.6961897

input -> 0.1x=0.5x-1.6
0.1x=0.5x-1.6 > 0.99999994
0.01x+0.98=0.5x > 0.61648583
0.8x-(0.01x-2)=0.03x-2 > 0.5868359
0.6x+0.4(3x-2)=4.6 > 0.5778694
0.6(3x+5)=2x-4.2 > 0.564164

input -> 2(x+5)=6
2(x+5)=6 > 0.9999999
2x+15=-x > 0.87820643
2x+1=x+7 > 0.8763264
2(x+1)=3x+4 > 0.72234654
2(x+3)=x+11 > 0.722071

input -> -4(x+5)=6
-4x+5=-11 > 0.956455
-3(4-x)=x > 0.8782562
-10x-9=-5 > 0.76801944
```

```
-10x+6=-5 > 0.7587959
-2x-3=6x+13 > 0.6597776

mode: E2D size: 22x200 layer: 4
input -> x-3=7
7+x=3 > 0.6438895
x-4=7 > 0.6387203
7x-3=9+4x > 0.55196625
16-3x=4x-2(5x+1) > 0.5509103
5-x=4 > 0.53452253

input -> f{4}{3}x-9=-10
f{4}{3}x-9=-10 > 0.9999999
f{3}{2}x-6=-5x > 0.7531116
f{7}{3}-7x=-3x > 0.6775274
-f{6}{7}+4x=2x > 0.5768339
f{4}{7}x-0.2x=-f{3}{8} > 0.5318122

input -> 0.1x=0.5x-1.6
0.1x=0.5x-1.6 > 0.9999999
7.2x-4=2.2x+9 > 0.5121792
0.11x-1.1=0.3x+0.61 > 0.43585777
2(x+4)=5(x-1) > 0.43488857
9-2(x-4)=3x+7 > 0.43163806

input -> 2(x+5)=6
2(x+5)=6 > 1.0
2x+15=-x > 0.6960634
13+4x=25 > 0.59619397
5x=8x+24 > 0.5613497
4x+2=3 > 0.560486

input -> -4(x+5)=6
-3(4-x)=x > 0.7466616
-4x+5=-11 > 0.71639407
4x+7=x-11 > 0.61074364
5x=2(x-9) > 0.59230787
-2(x+8)=6x-6 > 0.5850883

mode: E2D size: 22x200 layer: 8
input -> x-3=7
x-4=7 > 0.85924745
x+5=8 > 0.5697388
7+x=3 > 0.52871746
```

```
5-x=4 > 0.49663222
x+2=2 > 0.41224614

input -> f{4}{3}x-9=-10
f{4}{3}x-9=-10 > 0.99999994
f{3}{2}x-6=-5x > 0.817216
0.5(x-9)=-1.3x > 0.75615835
f{7}{3}-7x=-3x > 0.7523141
f{10}{9}x-3=-5 > 0.6933126

input -> 0.1x=0.5x-1.6
0.1x=0.5x-1.6 > 0.99999994
f{1}{2}x-6=-1 > 0.7779732
f{1}{2}x-4=11 > 0.7494417
f{5}{2}x+2=2x > 0.7179031
7.2x-4=2.2x+9 > 0.704125

input -> 2(x+5)=6
2(x+5)=6 > 0.99999994
-7x+10=9 > 0.629566
5x+7=9 > 0.51023567
2x+1=x+7 > 0.50873363
3x+5=7 > 0.48993105

input -> -4(x+5)=6
9x-8=5x+6 > 0.50285524
-10x+6=-5 > 0.45350274
-3(4-x)=x > 0.4236934
0.2(x+6)=0.5x-0.6 > 0.39269254
f{1}{3}x+0.5x=f{10}{9} > 0.38043457

mode: E2D size: 22x500 layer: 1
input -> x-3=7
x-4=7 > 0.95358276
x-5=5 > 0.9351595
x+5=8 > 0.7897098
x+2=2 > 0.78310066
3x=15 > 0.56113315

input -> f{4}{3}x-9=-10
f{4}{3}x-9=-10 > 0.99999976
f{3}{2}x-6=-5x > 0.95841056
f{7}{3}-7x=-3x > 0.9493747
f{10}{9}x-3=-5 > 0.9372679
```

```
f{4}{3}x-f{1}{3}=1 > 0.68089604

input -> 0.1x=0.5x-1.6
0.1x=0.5x-1.6 > 0.9999997
0.4x+0.6=0.2 > 0.6361535
0.3x+1.7=0.2 > 0.6258646
0.01x+0.98=0.5x > 0.6201839
0.2(x+6)=0.5x-0.6 > 0.6017087

input -> 2(x+5)=6
2(x+5)=6 > 1.0
2x+15=-x > 0.8271772
2x+1=x+7 > 0.82373536
2(4x+f{1}{2})=f{3}{4}(8x-12) > 0.6602281
2(3x-5)=3(x+2)+4(2x-1) > 0.6382477

input -> -4(x+5)=6
-4x+5=-11 > 0.95100325
-3(4-x)=x > 0.8223118
-10x-9=-5 > 0.73558027
-10x+6=-5 > 0.7320987
-4x+9=1 > 0.60368705

mode: E2D size: 22x500 layer: 4
input -> x-3=7
x-4=7 > 0.7856046
7+x=3 > 0.69112355
5-x=4 > 0.5967483
x-5=5 > 0.5130147
7x-3=9+4x > 0.4606765

input -> f{4}{3}x-9=-10
f{4}{3}x-9=-10 > 1.0000001
f{7}{3}-7x=-3x > 0.83058256
f{3}{2}x-6=-5x > 0.8218114
-f{9}{10}x+9=6 > 0.5694117
-f{6}{7}+4x=2x > 0.56142735

input -> 0.1x=0.5x-1.6
0.1x=0.5x-1.6 > 0.9999999
1.7x-0.7=1.6x-0.5 > 0.47243303
7.2x-4=2.2x+9 > 0.45301583
1.1x+0.6=0.85x-1.15 > 0.44245026
0.11x-1.1=0.3x+0.61 > 0.43595606
```

```
input -> 2(x+5)=6
2(x+5)=6 > 0.9999999
2x+15=-x > 0.5773586
2x+1=x+7 > 0.53703153
13+4x=25 > 0.52427554
5x=8x+24 > 0.5211626

input -> -4(x+5)=6
2x+5=8x+4 > 0.6925208
-4x+5=-11 > 0.67323875
3x+5=-x+2 > 0.6688769
-3(4-x)=x > 0.6653332
-10x+6=-5 > 0.59487367

mode: E2D size: 22x500 layer: 8
input -> x-3=7
x-4=7 > 0.9142918
7+x=3 > 0.7069854
5-x=4 > 0.4058542
-3x=-27 > 0.3466531
f{3}{7}x-3=-5 > 0.33707008

input -> f{4}{3}x-9=-10
f{4}{3}x-9=-10 > 0.9999999
f{7}{3}-7x=-3x > 0.8096655
f{3}{2}x-6=-5x > 0.80748725
f{10}{9}x-3=-5 > 0.784064
-f{9}{10}x+9=6 > 0.5998565

input -> 0.1x=0.5x-1.6
0.1x=0.5x-1.6 > 0.9999999
1.7x-0.7=1.6x-0.5 > 0.5260144
3.3x+1.4=0.6x-1.3 > 0.486443
f{10}{3}x-4=2 > 0.4502763
0.11x-1.1=0.3x+0.61 > 0.44960168

input -> 2(x+5)=6
2(x+5)=6 > 0.99999994
3x-7=-16 > 0.45176476
-10x-5=8 > 0.4313711
-10x+6=-5 > 0.4190701
-2(x+8)=6x-6 > 0.40451998
```

```
input -> -4(x+5)=6
9x-8=5x+6 > 0.7264374
-10x+6=-5 > 0.69968903
-3(4-x)=x > 0.5661228
2x+5=8x+4 > 0.5191195
-2(x+6)=5(x-4) > 0.4562482

mode: E2D size: 22x1000 layer: 1
input -> x-3=7
x-4=7 > 0.94222766
x-5=5 > 0.91883606
x+2=2 > 0.7476988
x+5=8 > 0.7455914
x-8=-3 > 0.59697235

input -> f{4}{3}x-9=-10
f{4}{3}x-9=-10 > 0.9999995
f{7}{3}-7x=-3x > 0.9553429
f{3}{2}x-6=-5x > 0.9542278
f{10}{9}x-3=-5 > 0.9382514
f{4}{3}x+7=-2 > 0.67028546

input -> 0.1x=0.5x-1.6
0.1x=0.5x-1.6 > 0.99999976
0.5(x-9)=-1.3x > 0.66829365
0.01x+0.98=0.5x > 0.666116
0.38-0.18x=0.2 > 0.6644658
0.6x+0.4(3x-2)=4.6 > 0.6233738

input -> 2(x+5)=6
2(x+5)=6 > 1.0000001
2x+15=-x > 0.8697335
2x+1=x+7 > 0.8586826
2(4x+1)=3(3x-5) > 0.6613675
2(5-x)=3(2x-5)+7x > 0.6594957

input -> -4(x+5)=6
-4x+5=-11 > 0.9619249
-3(4-x)=x > 0.8356369
-10x+6=-5 > 0.75671685
-10x-9=-5 > 0.7534554
-4x+9=1 > 0.63603586

mode: E2D size: 22x1000 layer: 4
```

```
input -> x-3=7
x-4=7 > 0.7486824
7+x=3 > 0.6698163
5-x=4 > 0.55339676
7x-3=9+4x > 0.45680988
3x-7=-2x+9 > 0.45146835

input -> f{4}{3}x-9=-10
f{4}{3}x-9=-10 > 1.0000001
f{7}{3}-7x=-3x > 0.85660386
f{3}{2}x-6=-5x > 0.8443472
-f{7}{6}x-7=-8 > 0.6784762
-f{9}{10}x+9=6 > 0.64887583

input -> 0.1x=0.5x-1.6
0.1x=0.5x-1.6 > 0.9999998
1.1x+0.6=0.85x-1.15 > 0.5060016
f{10}{3}x-4=2 > 0.50477433
0.01x+0.98=0.5x > 0.48945397
7.2x-4=2.2x+9 > 0.4888657

input -> 2(x+5)=6
2(x+5)=6 > 1.0000002
2x+15=-x > 0.61277586
2x+1=x+7 > 0.5298991
13+4x=25 > 0.5053128
-7x+10=9 > 0.4853241

input -> -4(x+5)=6
-4x+5=-11 > 0.6981378
-3(4-x)=x > 0.6155417
3x+5=-x+2 > 0.55136484
-10x+6=-5 > 0.5324473
2x+5=8x+4 > 0.5307147

mode: E2D size: 22x1000 layer: 8
input -> x-3=7
x-4=7 > 0.8718207
7+x=3 > 0.5398261
x+5=8 > 0.36014855
5x+7=9 > 0.34951353
4x-7=8 > 0.3381323

input -> f{4}{3}x-9=-10
```

```
f{4}{3}x-9=-10 > 0.9999999
f{7}{3}-7x=-3x > 0.82738566
f{3}{2}x-6=-5x > 0.7808547
f{10}{9}x-3=-5 > 0.75452656
-f{7}{6}x-7=-8 > 0.6444054

input -> 0.1x=0.5x-1.6
0.1x=0.5x-1.6 > 0.99999994
2(x+4)=5(x-1) > 0.46192527
-f{8}{7}x-5=2 > 0.41125873
0.8x-(0.01x-2)=0.03x-2 > 0.40646908
1.1x+0.6=0.85x-1.15 > 0.39467183

input -> 2(x+5)=6
2(x+5)=6 > 1.0000001
2x+1=x+7 > 0.36391753
3x-7=-16 > 0.36154145
2(x+4)=5(x-1) > 0.3344465
3x+2(5-x)=6 > 0.33249152

input -> -4(x+5)=6
9x-8=5x+6 > 0.6988591
-10x+6=-5 > 0.6161238
-3(4-x)=x > 0.47893324
2(x+5)=6 > 0.46710303
2x+5=8x+4 > 0.4185045

mode: SkipConnect size: 22x200 layer: 1
input -> x-3=7
x-4=7 > 0.9999608
x-5=5 > 0.9999327
x+2=2 > 0.9997291
x+5=8 > 0.99969065
x-8=-3 > 0.99945056

input -> f{4}{3}x-9=-10
f{4}{3}x-9=-10 > 1.0
f{3}{2}x-6=-5x > 0.99996567
f{7}{3}-7x=-3x > 0.99996275
f{10}{9}x-3=-5 > 0.9999544
f{1}{2}(4x-8)=5x+2 > 0.99960756

input -> 0.1x=0.5x-1.6
0.1x=0.5x-1.6 > 0.9999998
```

```
0.6(3x+5)=2x-4.2 > 0.99967533
0.4x+0.6=0.2 > 0.99958724
0.3x+1.7=0.2 > 0.99958664
0.11x-1.1=0.3x+0.61 > 0.9995676

input -> 2(x+5)=6
2(x+5)=6 > 0.9999998
2x+15=-x > 0.99987996
2x+1=x+7 > 0.9998721
2(4x+1)=3(3x-5) > 0.99954844
2(x+3)=8x+4 > 0.9995383

input -> -4(x+5)=6
-4x+5=-11 > 0.9999555
-3(4-x)=x > 0.99987245
-10x+6=-5 > 0.99976254
-10x-9=-5 > 0.99975955
-5(x-1)=-7(x+3) > 0.9995558

mode: SkipConnect size: 22x200 layer: 4
input -> x-3=7
x-4=7 > 0.9993723
x-5=5 > 0.9989061
x+2=2 > 0.99580735
x+5=8 > 0.9952584
x-8=-3 > 0.9914947

input -> f{4}{3}x-9=-10
f{4}{3}x-9=-10 > 1.0000001
f{3}{2}x-6=-5x > 0.99944305
f{7}{3}-7x=-3x > 0.9994031
f{10}{9}x-3=-5 > 0.9992831
f{1}{2}(4x-8)=5x+2 > 0.9937578

input -> 0.1x=0.5x-1.6
0.1x=0.5x-1.6 > 0.99999994
0.6(3x+5)=2x-4.2 > 0.9948791
0.4x+0.6=0.2 > 0.99342984
0.3x+1.7=0.2 > 0.9934135
0.8x-(0.01x-2)=0.03x-2 > 0.99320745

input -> 2(x+5)=6
2(x+5)=6 > 1.0000001
2x+15=-x > 0.99805224
```

```
2x+1=x+7 > 0.99793
2(4x+1)=3(3x-5) > 0.9928722
2(x+3)=8x+4 > 0.9926963

input -> -4(x+5)=6
-4x+5=-11 > 0.99926853
-3(4-x)=x > 0.99795955
-10x+6=-5 > 0.9962721
-10x-9=-5 > 0.99624175
-5(x-1)=-7(x+3) > 0.9928169

mode: SkipConnect size: 22x200 layer: 8
input -> x-3=7
x-4=7 > 0.99746716
x-5=5 > 0.99554414
x+2=2 > 0.9840825
x+5=8 > 0.9823385
x-8=-3 > 0.96806246

input -> f{4}{3}x-9=-10
f{4}{3}x-9=-10 > 0.9999998
f{3}{2}x-6=-5x > 0.9977706
f{7}{3}-7x=-3x > 0.9976454
f{10}{9}x-3=-5 > 0.99721265
f{9}{2}+0.3x=-f{9}{2} > 0.9755826

input -> 0.1x=0.5x-1.6
0.1x=0.5x-1.6 > 0.9999998
0.6(3x+5)=2x-4.2 > 0.98000866
0.4x+0.6=0.2 > 0.9741379
0.3x+1.7=0.2 > 0.97401446
0.8x-(0.01x-2)=0.03x-2 > 0.9738555

input -> 2(x+5)=6
2(x+5)=6 > 1.0000001
2x+15=-x > 0.9921457
2x+1=x+7 > 0.99164784
2(4x+1)=3(3x-5) > 0.9723573
2(3x-5)=3(x+2)+4(2x-1) > 0.97182924

input -> -4(x+5)=6
-4x+5=-11 > 0.9969909
-3(4-x)=x > 0.9919083
-10x+6=-5 > 0.9856282
```

```
-10x-9=-5 > 0.98561376
-5x=20 > 0.97107166

mode: SkipConnect size: 22x500 layer: 1
input -> x-3=7
x-4=7 > 0.99987143
x-5=5 > 0.9997851
x+5=8 > 0.9995236
x+2=2 > 0.99944633
x-8=-3 > 0.9988501

input -> f{4}{3}x-9=-10
f{4}{3}x-9=-10 > 0.99999994
f{3}{2}x-6=-5x > 0.99989796
f{7}{3}-7x=-3x > 0.9998935
f{10}{9}x-3=-5 > 0.99986035
f{7}{4}+3x=-1 > 0.99905956

input -> 0.1x=0.5x-1.6
0.1x=0.5x-1.6 > 1.0000002
0.11x-1.1=0.3x+0.61 > 0.99904037
0.6x+0.4(3x-2)=4.6 > 0.9989626
0.5(x-9)=-1.3x > 0.99895513
0.8x-(0.01x-2)=0.03x-2 > 0.99891675

input -> 2(x+5)=6
2(x+5)=6 > 0.99999994
2x+15=-x > 0.9996994
2x+1=x+7 > 0.99968576
2(x+4)=5(x-1) > 0.9989838
2(5-x)=3(2x-5)+7x > 0.99895805

input -> -4(x+5)=6
-4x+5=-11 > 0.9998899
-3(4-x)=x > 0.9996674
-10x+6=-5 > 0.9994921
-10x-9=-5 > 0.99945056
-4x+9=1 > 0.9989111

mode: SkipConnect size: 22x500 layer: 4
input -> x-3=7
x-4=7 > 0.99785835
x-5=5 > 0.9964146
x+5=8 > 0.992463
```

```
x+2=2 > 0.991218

input -> f{4}{3}x-9=-10
f{4}{3}x-9=-10 > 0.9999999
f{3}{2}x-6=-5x > 0.9983822
f{7}{3}-7x=-3x > 0.99830186
f{10}{9}x-3=-5 > 0.99782133
f{7}{4}+3x=-1 > 0.9853469

input -> 0.1x=0.5x-1.6
0.1x=0.5x-1.6 > 1.0000002
0.11x-1.1=0.3x+0.61 > 0.9850222
0.6x+0.4(3x-2)=4.6 > 0.98371047
0.5(x-9)=-1.3x > 0.9833958
0.8x-(0.01x-2)=0.03x-2 > 0.9829791

input -> 2(x+5)=6
2(x+5)=6 > 0.99999994
2x+15=-x > 0.9952541
2x+1=x+7 > 0.9950551
2(x+4)=5(x-1) > 0.98401433
2(5-x)=3(2x-5)+7x > 0.98363835

input -> -4(x+5)=6
-4x+5=-11 > 0.99834
-3(4-x)=x > 0.9950138
-10x+6=-5 > 0.99222285
-10x-9=-5 > 0.99158734
-4x+9=1 > 0.98335236

mode: SkipConnect inputsize: 22x500 layer: 8
input -> x-3=7
x-4=7 > 0.9908074
x-5=5 > 0.9840723
x+5=8 > 0.97010446
x+2=2 > 0.96485144
x-8=-3 > 0.93049264

input -> f{4}{3}x-9=-10
f{4}{3}x-9=-10 > 0.99999976
f{3}{2}x-6=-5x > 0.9936152
f{7}{3}-7x=-3x > 0.99314535
f{10}{9}x-3=-5 > 0.9913904
f{7}{4}+3x=-1 > 0.9447946
```

```
input -> 0.1x=0.5x-1.6
0.1x=0.5x-1.6 > 1.0000001
0.11x-1.1=0.3x+0.61 > 0.94313234
0.6x+0.4(3x-2)=4.6 > 0.93787944
0.5(x-9)=-1.3x > 0.9359052
0.8x-(0.01x-2)=0.03x-2 > 0.93485516

input -> 2(x+5)=6
2(x+5)=6 > 0.9999997
2x+15=-x > 0.9814998
2x+1=x+7 > 0.98066026
2(x+4)=5(x-1) > 0.93875295
2(5-x)=3(2x-5)+7x > 0.9380875

input -> -4(x+5)=6
-4x+5=-11 > 0.99358886
-3(4-x)=x > 0.98175734
-10x+6=-5 > 0.97098434
-10x-9=-5 > 0.9683183
-4x+9=1 > 0.93842256

mode: SkipConnect size: 22x1000 layer: 1
input -> x-3=7
x-4=7 > 0.9995117
x-5=5 > 0.999067
x+5=8 > 0.99828815
x+2=2 > 0.997988
x-8=-3 > 0.99751353

input -> f{4}{3}x-9=-10
f{4}{3}x-9=-10 > 1.0000002
f{3}{2}x-6=-5x > 0.9996954
f{10}{9}x-3=-5 > 0.9996778
f{7}{3}-7x=-3x > 0.9996517
f{7}{4}+8x=-f{4}{3} > 0.9979254

input -> 0.1x=0.5x-1.6
0.1x=0.5x-1.6 > 0.99999964
0.3x+1.7=0.2 > 0.99786717
0.5(x-9)=-1.3x > 0.9978409
0.11x-1.1=0.3x+0.61 > 0.9978199
0.4x+0.6=0.2 > 0.9976466
```

```
input -> 2(x+5)=6
2(x+5)=6 > 1.0
2x+15=-x > 0.9992464
2x+1=x+7 > 0.9990841
2(x+4)=5(x-1) > 0.99761903
2(5-x)=3(2x-5)+7x > 0.99759763

input -> -4(x+5)=6
-4x+5=-11 > 0.9994779
-3(4-x)=x > 0.9991171
-10x+6=-5 > 0.9987582
-10x-9=-5 > 0.9984923
-4x+9=1 > 0.99770147

mode: SkipConnect size: 22x1000 layer: 4
input -> x-3=7
x-4=7 > 0.99194837
x-5=5 > 0.9850264
x+5=8 > 0.9727686
x+2=2 > 0.96837586
x-8=-3 > 0.96346414

input -> f{4}{3}x-9=-10
f{4}{3}x-9=-10 > 1.0000004
f{3}{2}x-6=-5x > 0.9949234
f{10}{9}x-3=-5 > 0.994846
f{7}{3}-7x=-3x > 0.99448276
f{7}{4}+8x=-f{4}{3} > 0.96812683

input -> 0.1x=0.5x-1.6
0.1x=0.5x-1.6 > 0.99999934
0.5(x-9)=-1.3x > 0.9674616
0.3x+1.7=0.2 > 0.96616524
0.11x-1.1=0.3x+0.61 > 0.96519846
0.38-0.18x=0.2 > 0.96354043

input -> 2(x+5)=6
2(x+5)=6 > 0.99999994
2x+15=-x > 0.98820996
2x+1=x+7 > 0.98550105
2(x+4)=5(x-1) > 0.96330446
2(5-x)=3(2x-5)+7x > 0.9632084

input -> -4(x+5)=6
```

```
-4x+5=-11 > 0.99144673
-3(4-x)=x > 0.9864846
-10x+6=-5 > 0.98037875
-10x-9=-5 > 0.9759923
-4x+9=1 > 0.9651148

mode: SkipConnect size: 22x1000 layer: 8
input -> x-3=7
x-4=7 > 0.963662
x-5=5 > 0.9338983
x+5=8 > 0.8888245
x+2=2 > 0.8710501
x-8=-3 > 0.86731726

input -> f{4}{3}x-9=-10
f{4}{3}x-9=-10 > 0.9999998
f{10}{9}x-3=-5 > 0.9777564
f{3}{2}x-6=-5x > 0.976848
f{7}{3}-7x=-3x > 0.97645557
f{7}{4}+8x=-f{4}{3} > 0.88040847

input -> 0.1x=0.5x-1.6
0.1x=0.5x-1.6 > 1.0000001
0.5(x-9)=-1.3x > 0.8799058
0.3x+1.7=0.2 > 0.8660037
0.38-0.18x=0.2 > 0.8646081
0.8x-(0.01x-2)=0.03x-2 > 0.8599009

input -> 2(x+5)=6
2(x+5)=6 > 1.0000004
2x+15=-x > 0.9525059
2x+1=x+7 > 0.94002306
2(x+4)=5(x-1) > 0.865899
2(5-x)=3(2x-5)+7x > 0.8627077

input -> -4(x+5)=6
-4x+5=-11 > 0.95918113
-3(4-x)=x > 0.949258
-10x+6=-5 > 0.9239827
-10x-9=-5 > 0.90235525
-4x+9=1 > 0.8695013

mode: BiDirection size: 22x200 layer: 1
```

```
input -> x-3=7
x-4=7 > 0.99999756
x+3=-8 > 0.99999684
x-8=-3 > 0.99999624
x+5=8 > 0.9999952
x+2=2 > 0.9999951

input -> f{4}{3}x-9=-10
f{4}{3}x-9=-10 > 1.0
f{7}{4}+8x=-f{4}{3} > 0.99999976
f{1}{3}(3x-12)=2x+8 > 0.9999997
f{3}{4}x-1=5 > 0.9999997
f{3}{7}x-3=-5 > 0.99999964

input -> 0.1x=0.5x-1.6
0.1x=0.5x-1.6 > 1.0
0.11x-1.1=0.3x+0.61 > 0.99999905
0.5(x-9)=-1.3x > 0.99999845
0.01x+0.98=0.5x > 0.99999845
0.2(x+6)=0.5x-0.6 > 0.99999845

input -> 2(x+5)=6
2(x+5)=6 > 1.0
2(x+4)=5(x-1) > 0.9999997
2(3x-5)=3(x+2)+4(2x-1) > 0.9999993
2(x+3)=8x+4 > 0.9999992
2x+1=5 > 0.9999991

input -> -4(x+5)=6
-2(x+6)=5(x-4) > 0.99999905
-f{1}{3}(9x+24)+6(f{1}{3}x-1)=f{1}{4}(16x+8)+3x-1 > 0.9999986
-3(2x-8)=4(x+1) > 0.99999833
-10x+6=-5 > 0.9999983
-5(x-1)=-7(x+3) > 0.9999983

mode: BiDirection size: 22x500 layer: 1
input -> x-3=7
x-4=7 > 0.99995583
x+3=-8 > 0.9999467
x-8=-3 > 0.9999465
x+5=13 > 0.999936
x+5=8 > 0.9999135

input -> f{4}{3}x-9=-10
```

```
f{4}{3}x-9=-10 > 1.0
f{3}{4}x-1=5 > 0.99999106
f{10}{3}x-4=2 > 0.9999885
f{4}{3}x+7=-2 > 0.9999882
f{2x-1}{3}-f{4x-3}{2}=-4 > 0.9999881

input -> 0.1x=0.5x-1.6
0.1x=0.5x-1.6 > 1.0
0.11x-1.1=0.3x+0.61 > 0.9999852
0.5(x-9)=-1.3x > 0.9999808
0.2(x+6)=0.5x-0.6 > 0.99997926
0.01x+0.98=0.5x > 0.99997735

input -> 2(x+5)=6
2(x+5)=6 > 0.99999964
2(x+4)=5(x-1) > 0.9999855
2(5-x)=3(2x-5)+7x > 0.99997926
2(x+3)=8x+4 > 0.99997824
2(x+1)=3x+4 > 0.9999778

input -> -4(x+5)=6
-2(x+6)=5(x-4) > 0.9999838
-3(2x-8)=4(x+1) > 0.9999718
-2(x-5)=9x-8 > 0.99997026
-2(x+8)=6x-6 > 0.99996954
-10x+6=-5 > 0.9999664

mode: BiDirection size: 22x1000 layer: 1
input -> x-3=7
x-4=7 > 0.99962753
x+3=-8 > 0.99959344
x-8=-3 > 0.99951357
x+5=13 > 0.9994478
x+5=8 > 0.999257

input -> f{4}{3}x-9=-10
f{4}{3}x-9=-10 > 1.0000002
f{3}{4}x-1=5 > 0.9999203
f{4}{3}x-f{1}{3}=1 > 0.99989825
f{4}{7}x-0.2x=-f{3}{8} > 0.9998977
f{2x-1}{3}-f{4x-3}{2}=-4 > 0.9998956

input -> 0.1x=0.5x-1.6
0.1x=0.5x-1.6 > 1.0000001
```

```
0.11x-1.1=0.3x+0.61 > 0.9998635
0.2(x+6)=0.5x-0.6 > 0.9998056
0.01x+0.98=0.5x > 0.9997614
0.5(x-9)=-1.3x > 0.9997599

input -> 2(x+5)=6
2(x+5)=6 > 0.9999997
2(x+4)=5(x-1) > 0.999883
2(x+1)=3x+4 > 0.9998496
2(3x-5)=3(x+2)+4(2x-1) > 0.9998377
2(x+3)=8x+4 > 0.9998265

input -> -4(x+5)=6
-2(x+6)=5(x-4) > 0.999892
-3(2x-8)=4(x+1) > 0.99982303
-f{1}{3}(9x+24)+6(f{1}{3}x-1)=f{1}{4}(16x+8)+3x-1 > 0.99979335
-2(x+8)=6x-6 > 0.99978095
-0.4(f{5}{2}x+12)=-3 > 0.9997515
```

参考文献

[1] pagesComputer Science Department, Stanford University, Stanford, CA 94305 (2016).

[2] TomasMikolovBag of Tricks for Efficient Text Classification (2016).

[3] HarishKarnickSCDV : Sparse Composite Document Vectors using soft clustering over distributional representations (2016).

[4] JeffreyDeanEfficient Estimation of Word Representations in Vector Space (2013).

[5] Quoc V.LeSequence to Sequence Learning with Neural Networks (2014).

[6] JiweiLiGlyce: Glyph-vectors for Chinese Character Representations (2019).

[7] year 第 6 回学習指導基本調査 DATA BOOK（高校版) (2016).

[8] 松尾豊 The 28th Annual Conference of the Japanese Society for Artificial Intelligence, (2014).

[9] title 全国学力・学習状況調査.

[10] pagesThe 31st Annual Conference of the Japanese Society for Artificial Intelligence (2017).

[11] yearO.REILLY (2018).

[12] pages 言語処理学会 第 24 回年次大会 発表論文集 (2018).

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．また研究が行き詰まると，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．そしてこの研究では自然言語処理の奥深さをただただ感じました．そんな時，常に参考になり研究の日々を共に過ごしたゼロから作る DeepLearning 2 自然言語処理編 [11] には山ほど助けられました．Python から scala に書き換えながら理解を進めることができ芯から理解することができたように思います．

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心から感謝と未来の活躍を祈るとともに必ず，この経験を糧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．