Vol.2017-NL-231 No.13
Vol.2017-SLP-116 No.13
2017/5/15

遷移型句構造解析に基づく論文PDF中の数式XML解析

澤井 裕一郎[1,a)]　進藤 裕之[1,b)]　松本 裕治[1,c)]

概要：数式は科学技術論文において多くの情報を担う重要な要素であり，論文の意味理解や高度な検索のためには，論文中の数式の高精度な構造解析が求められる．本研究では，科学技術論文の PDF ファイルに含まれる数式を対象に，数式の構造記述に特化した XML の一種である MathML 形式を予測するタスクに取り組む．特に，PDF ファイルから抽出した文字・図形情報の系列を入力として，XML の木構造を同定する句構造解析の問題として捉え，従来の遷移型句構造解析の手法を拡張して適用する．そして，医学分野の論文に含まれる数式データを用いて評価実験を行い，解析性能を報告する．

1. はじめに

日々出版される科学技術論文の量は膨大であり，最新の論文を追い続ける必要がある研究者の負担を軽減するために，計算機による論文からの知識抽出・情報検索技術が求められている．科学技術論文は一般的に PDF（Portable Document Format）ファイルとして公開される．しかし，PDF は一般的に文や段落，節などの構造情報を保持していないため，そのままの状態では情報抽出や情報検索に使用することが困難である．したがって，論文 PDF に含まれる段落構成や図表・数式等を解析し，XML のような構造化された形式に変換する技術が必要である．論文の構成要素の中でも特に数式は，科学技術論文において多くの情報を担う重要な要素である．実際，NTCIR-12 MathIR タスク [12] のように，論文中の数式を対象とした情報検索に関する Shared Task が開催されている．そこで本研究では，PDF に含まれる数式を，数式の構造記述に特化した XML の一種である MathML 形式に変換するタスク（数式 XML 解析タスク）に取り組む．PDF において，数式は文字や直線の描画命令の系列で表現されている．したがって，上付き・下付き文字の区別や，sin や cos などのトークン単位を認識することができない．一方，MathML 形式の数式では，上付き，下付き文字の区別や，トークンの単位が XML タグによって明示的に表現されており，それらがどのように数式全体を構成しているのかが明らかである．タスクの概要を図 1 に示す．

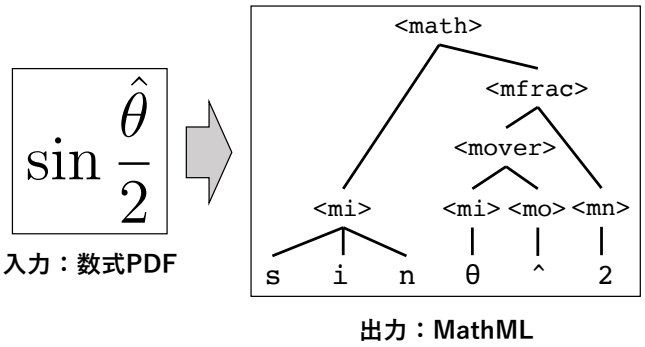

図 1　数式 XML 解析タスク

数式 XML 解析タスクに取り組むため，本研究では医学分野のオンライン論文アーカイブである PMC のデータを用いる．PMC では，JATS 形式（科学技術論文の構造を表現するための XML の一種）で論文を投稿することが求められているため，JATS 形式のテキストファイルと，それを変換した PDF ファイルの両方を入手可能である．我々は，PMC から取得した論文の JATS 形式から数式部分（MathML 形式）のみを抽出し，それを PDF に変換することで，数式 PDF と MathML 形式のペアのデータセットを作成した．そして，このデータセットを用いて数式 XML 解析モデルの評価実験を行う．

Deng ら [5] は，画像キャプション生成の手法 [11] を応用して，数式画像から数式の LaTeX 形式を予測する手法を提案した．この手法では，畳み込みニューラルネットワーク（Convolutional Neural Network; CNN）を用いて数式画像を読み取り，リカレントニューラルネットワーク（Recurrent Neural Network; RNN）を用いて出力を 1 トークンずつ系列として生成していく．しかしながら，この手法では，数式 PDF に含まれる文字を重複して生成したり，逆に一部を生成しないという問題点がある．また，出力が

[1] 奈良先端科学技術大学院大学
Nara Institute of Science and Technology
[a)] sawai.yuichiro.sn0@is.naist.jp
[b)] shindo@is.naist.jp
[c)] matsu@is.naist.jp

木構造であるという制約を考慮せずに系列として生成するため，生成されたトークン系列が木構造として解釈できない場合がある．

そこで本研究では，数式 XML 解析タスクを，PDF を画像に変換するのではなく，PDF ファイルから直接抽出した文字・図形情報の系列に対する句構造解析として捉え，遷移型句構造解析に基づく解析手法を提案する．この手法の利点として，数式 PDF 中に含まれる文字・図形情報を順次過不足なく利用することにより，XML 生成時の文字の重複や欠損の問題を回避できる点や，句構造解析の手法を適用するため必ず木構造の MathML を生成できる点が挙げられる．

PMC データを使った実験により，PDF から直接抽出した文字・図形情報を使うことと，遷移型句構造解析の手法を適用することで，従来手法に比べて解析精度が大幅に向上することを示す．

2. 数式 XML 解析データセットの構築

MathML

MathML は，数式構造を表現するための XML の一種である．図 1 に例を挙げる．<mi>，<mo>，<mn>はそれぞれ識別子，演算子，数を意味するタグである．<mover>は上付き文字，<mfrac>は分数を意味するタグである．<math>は MathML の根にあたるタグである．

PMC データの収集

本研究で行う実験のために，数式 PDF と MathML のペアのデータセットを構築する．まず，PMC[*1]で公開されている JATS 形式による論文の XML ファイルを取得し，<math>タグで囲まれた部分のみを抽出する．数式 PDF の表示と無関係な<annotation>タグと<semantics>タグを削除し，既存のツール[*2] を用いて，表示のための情報を持つ XSL-FO 形式に変換する．そして，AHFormatter v6.4[*3]により XSL-FO 形式を数式 PDF に変換する．

MathML の正規化

取得した MathML は，そのままでは，スタイル情報などの数式の構造以外の情報を多く含み，論文 PDF の構造解析の目的では不必要なタグが多い．また，同じ構造の数式に対して複数のタグの付け方が存在する場合があり，曖昧性が高い．そこで，スタイル情報の除去と曖昧性の低減のために，予測対象の MathML を正規化する．まず，MathMLCan[*4]を用いて正規化を行う．このツールは，タグの属性の削除，不必要な<mrow>タグ（複数のタグが一まとまりであることを表現）の除去などを行う．属性の除去により，スタイルに関する情報の大部分を取り除くことができる．更に，連続する複数の<mi>タグを 1 つにまとめるなどの正規化を追加で行う．正規化の例を図 2 に示す．

```
<math>
  <mi mathsize="9pt">s</mi>
  <mi mathsize="9pt">i</mi>
  <mi mathsize="9pt">n</mi>
  <mfrac>
    <mrow>
      <mover>
        <mi>θ</mi>
        <mo>^</mo>
      </mover>
    </mrow>
    <mrow>
      <mn>2</mn>
    </mrow>
  </mfrac>
</math>
```

```
<math>
  <mi>sin</mi>
  <mfrac>
    <mover>
      <mi>θ</mi>
      <mo>^</mo>
    </mover>
    <mn>2</mn>
  </mfrac>
</math>
```

図 2 MathML 正規化の例

表 1 数式 XML 解析データセットの統計量

訓練データ	開発データ	評価データ
219,064	3,000	3,000

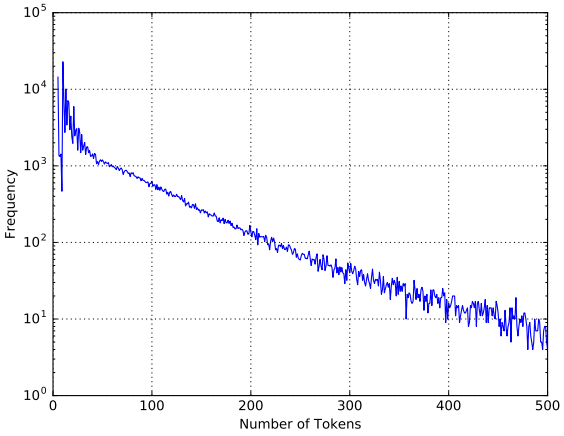

図 3 MathML のトークン数の分布（トークン数 500 まで）

データセットの統計量

最終的に得られた数式 XML 解析データセットの統計量を表 1 に示す．また，MathML のトークン数の分布を図 3 に示す．ただし，開きタグ，閉じタグはそれぞれ 1 トークンとして扱い，MathML の葉ノードにあたる文字列は 1 文字を 1 トークンとした．例えば，図 2 の正規化後の MathML は，開きタグが 7 トークン，閉じタグが 7 トークン，文字が 6 トークンの，合計 20 トークンにトークン化される．

3. 数式画像の Encoder-Decoder モデル

Deng ら [5] は，画像からのマークアップ言語（LaTeX や HTML）生成のタスクに対して，多層 CNN による画像 Encoder と，Attention 機構付き RNN による Decoder の組み合わせから成る，Encoder-Decoder モデルに基づく手法（WYGIWYS）を提案した．この手法により，数式画像から数式 LaTeX を生成するタスクにおいて，最高精度の商

*1 ftp://ftp.ncbi.nlm.nih.gov/pub/pmc/oa_package
*2 https://github.com/ncbi/JATSPreviewStylesheets/blob/master/xslt/main/jats-xslfo.xsl
*3 http://www.antenna.co.jp/AHF/
*4 https://github.com/michal-ruzicka/MathMLCan

表 2 CNN のネットワーク構成（c：出力チャネル数，k：カーネル幅，s：スライド幅，p：パディング幅，po：pooling 幅）

出力
Batch Normalization
畳み込み (c:512, k:3x3, s:1x1, p:1x1)
Max-Pooling (po:1x2, s:1x2, p:0x0)
Batch Normalization
畳み込み (c:512,k:3x3, s:1x1, p:1x1)
Max-Pooling (po:2x1, s:2x1, p:0x0)
畳み込み (c:256, k:3x3, s:1x1, p:1x1)
Batch Normalization
畳み込み (c:256, k:3x3, s:1x1, p:1x1)
Max-Pooling (po:2x2, s:2x2, p:0x0)
畳み込み (c:128, k:3x3, s1x1, p:1x1)
Max-Pooling (po:2x2, s:2x2, p:0x0)
畳み込み (c:64, k:3x3, s1x1, p:1x1)
入力

用数式 OCR（Optical Character Recognition）ソフトウェアである InftyReader [10] を上回る精度を報告している．

本研究では，ベースラインとして彼らの手法を用いる．即ち，画像に変換した数式 PDF[*5]を入力とし，2 節で述べた方法でトークン化された MathML のトークン列を出力する．以下に，彼らの手法の概要を説明する．

3.1 画像 Encoder

画像 Encoder は入力としてグレースケールの数式画像 $\mathbf{x} \in \mathbb{R}^{H \times W}$ を受け取る．ここで，H は画像の高さ，W は画像の幅である．$\mathbf{x}$ の各要素は，[-1, 1] の範囲にスケーリングされている．この数式画像に対して，画像認識において標準的に用いられている，畳み込み，Max-Pooling，Batch Normalization を順次適用するような多層 CNN を適用する．結果として，d 次元素性ベクトルのグリッド $\mathbf{V} \in \mathbb{R}^{H' \times W' \times d}$ を得る．ただし，H'，W' は，それぞれ Max-Pooling により縮小された画像の高さ，幅である．本研究では，ベースラインとして表 2 に示す Deng ら [5] が用いたものと同じ構成のネットワークを用いる．

画素 (i, j) に対応する素性ベクトル $\mathbf{V}_{ij}$ には画素の局所的な情報がエンコードされている．一方，数式の構造認識のためには，画素間の相対的な位置関係を考慮する必要がある．そこで，水平方向の位置関係を考慮した素性ベクトルを作るために，各 i 行目の素性ベクトル列 $\{\mathbf{V}_{ij}\}_{j=1}^{W'}$ に対して，双方向型 Long Short-Term Memory（LSTM）[6] を適用し，水平方向の大局的な情報をエンコードした素性ベクトルのグリッド $\tilde{\mathbf{V}} \in \mathbb{R}^{H' \times W' \times d}$ を得る．$\tilde{\mathbf{V}}$ の各行を一列に並べた d 次元素性ベクトル列 $\{\mathbf{h}_i^{IMG}\}_{i=1}^{H' \times W'}$ が画像 Encoder の出力である．

[*5] ImageMagick の convert −density 200 −quality 100 コマンドにより数式 PDF を PNG 形式に変換し，数式領域のみをクロッピングした．詳細は [5] 参照．

3.2 Attention 機構付き Decoder

Decoder は，Encoder（節 3.1 の画像 Encoder または後述する節 4.2 の PDF Encoder）が出力する素性ベクトル列 $\{\mathbf{h}_i\}_{i=1}^{N}$（前節の画像 Encoder を用いる場合は $N = H' \times W'$）を入力として，Attention 機構 [2] 付き RNN（LSTM を用いる）により，トークン列を生成する．

時刻 t において，LSTM の隠れ状態ベクトル $\mathbf{d}_t$ から，i 番目の素性ベクトル $\mathbf{h}_i$ に対する Attention 重み α_{it} が以下のように計算される．

$$e_{it} = \mathbf{h}_i^T \mathbf{d}_t$$
$$\alpha_{it} = \frac{\exp(e_{it})}{\sum_{i'=1}^{N} \exp(e_{i't})}$$

$\{\mathbf{h}_i\}_{i=1}^{N}$ の Attention 重みによる重み付き和を計算し，時刻 t における文脈ベクトル $\mathbf{c}_t$ を得る．

$$\mathbf{c}_t = \sum_{i=1}^{N} \alpha_{it} \mathbf{h}_i$$

$\mathbf{h}_t$ と $\mathbf{c}_t$ に基づき，時刻 t の出力トークンを softmax 関数で予測する．

$$\mathbf{o}_t = \tanh(\mathbf{W}_o[\mathbf{h}_t; \mathbf{c}_t])$$
$$\mathbf{y}_t = \mathrm{softmax}(\mathbf{W}_y \mathbf{o}_t)$$

ここで，$\mathbf{W}_o$，$\mathbf{W}_y$ はパラメータ行列である（バイアス項は省略している）．予測されたトークンの埋め込みベクトルが，次の時刻 $t+1$ における LSTM の入力である．

Deng ら [5] はこれに加えて，Input Feeding [8]（時刻 $t+1$ の入力埋め込みベクトルに $\mathbf{o}_t$ を結合）を行っている．我々の予備実験では Input Feeding による性能向上が見られなかったため，今回の実験では使用していない．

学習時は，各時刻 t における出力 $\mathbf{y}_t$ に対する交差エントロピー誤差関数を計算し，時刻 $1...T$（T はトークン列の長さ）での和を最小化するようにパラメータ最適化を行う．ただし，時刻 1 における LSTM への入力は<BOS>トークンの埋め込みベクトルとし，時刻 T での出力は<EOS>トークンとする．

4. 提案手法

Deng ら [5] の手法では，数式が画像として与えられることを前提としていた．しかし，今回の問題設定では，数式の PDF ファイルが利用可能であり，そこから文字・図形の情報を直接抽出することができる．表 3 に，図 1 の PDF から抽出した文字・図形情報の例を示す．

数式 XML から変換された PDF は，数式 XML 中の文字の順番と，PDF 中の文字や図形の順番が同じであることが期待できる．実際に，我々の構築したデータセットでは，変換元となる MathML に含まれる文字の順番と，変換先の PDF に含まれる文字・図形の順番は同じであった．

表 3 図 1 の数式 PDF から抽出した文字・図形情報の例．各行が 1 つの描画命令に対応する．背景が白の行は文字描画命令，灰色の行は図形描画命令を表す．1 つの文字描画命令は 1 つの文字トークンに対応する．図形描画命令は [MOVE_TO] から始まる 1 連のものが，1 つの図形トークンに対応する，

トークン番号	文字/図形描画命令	f_1	f_2	f_3	f_4	f_5	f_6
1	s	0.0	20.4	4.4	6.0	12.0	6.0
2	i	4.4	20.4	3.3	6.0	12.0	6.0
3	n	7.7	20.4	6.5	6.0	12.0	6.0
4	[MOVE_TO]	17.4	17.4				
4	[LINE_TO]	23.2	17.4				
5	θ	17.4	12.3	5.5	6.0	12.0	6.0
6	[MOVE_TO]	20.0	2.5				
6	[LINE_TO]	21.4	1.2				
6	[LINE_TO]	22.8	2.5				
7	2	17.4	28.6	5.9	6.0	12.0	6.0

したがって，本研究では，PDF から抽出した文字・図形情報の系列を入力として，MathML の木構造を出力する句構造解析の問題として捉える．PDF 変換ソフトウェアによっては，XML に含まれる文字の順序と，PDF に含まれる文字や図形の順序は異なる可能性があるが，そのような場合の取り扱いは今後の課題とする．

4.1 PDF からの文字・図形情報の抽出

前述のように，PDF ファイルは，内部的には文字・図形の描画命令の系列で構成されている．数式 PDF からこれらの命令列を抽出するために，我々は PDFBox v2.0.5[*6] を用いた．

図 1 の数式 PDF から抽出できる情報の例を，表 3 に示す．表 3 で示されているように，PDF の中身は，文字や図形の描画命令の系列となっており，それぞれが複数の実数値の情報 $\{f_i\}$ を持っている．例えば，文字であれば，$f_1, ..., f_6$ はそれぞれ，x 座標，y 座標，幅，高さ，フォントサイズ，フォントにおける空白の幅，を意味する．図形の描画命令は，[MOVE_TO]，[LINE_TO]，[CURVE_TO] の三種類が存在する．[MOVE_TO] は (f_1, f_2) から描画を開始することを表す．[LINE_TO] は前回の描画位置から，(f_1, f_2) まで直線を描画することを表す．[CURVE_TO] は前回の描画位置から，実数値 $(f_1, ..., f_6)$ で指定されるパラメータを持ったベジェ曲線を描画することを表す．[MOVE_TO], [LINE_TO] は 2 つの実数値を持ち，[CURVE_TO] は 6 つの実数値を持つ．これらの位置やサイズの情報は，PDF から MathML を予測する際に重要な特徴となり得るので，それぞれ訓練データ全体で平均 0，分散 1 になるように正規化し，入力素性として利用することとする．

これらの描画命令に対して，1 つの文字描画命令を 1 トークン，[MOVE_TO] から始まる一続きの図形描画命令を 1 トークンとして，トークン化する．文字トークンの集合を $\mathcal{C}$，図形トークンの集合を $\mathcal{D}$，$\mathcal{T} = \mathcal{C} \cup \mathcal{D}$ とすると，最終的に，数式 PDF からは文字・図形情報からなるトークン列 $\{t_i \in \mathcal{T}\}_{i=1}^{N}$ が抽出される．

[*6] https://pdfbox.apache.org/

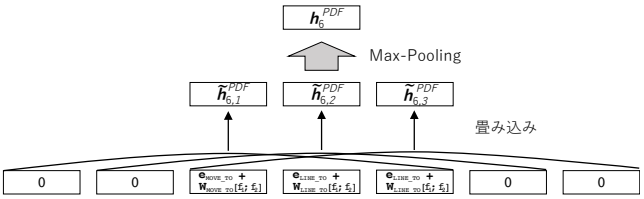

図 4 図形トークンに対する PDF Encoder の計算例（h_6^{PDF}）

4.2 PDF Encoder

次に，ニューラルネットワークを用いて，数式 PDF から抽出したトークン列 $\{t_i\}_{i=1}^{N}$ から特徴ベクトルを計算する PDF Encoder について説明する．

まず，i 番目のトークン t_i に対するベクトル表現 $\mathbf{h}_i^{PDF}$ を計算する．$t_i \in \mathcal{C}$ の場合，$\mathbf{h}_i^{PDF}$ は，文字の埋め込みベクトル $\mathbf{e}_c$ と，$f_1, ..., f_6$ を線形写像によって変換したものの和とする．すなわち，

$$\mathbf{h}_i^{PDF} = \mathbf{e}_c + \mathbf{W}_{\text{char}}[f_1; f_2; ...; f_6]$$

ただし，$\mathbf{W}_{\text{char}}$ は $d \times 6$ の実数値パラメータ行列である．

$t_i \in \mathcal{D}$ の場合，m_i を t_i が含む図形描画命令の個数（表 3 の例では，$m_4 = 2$，$m_6 = 3$）とし，j 番目の描画命令のベクトル表現 $\mathbf{h}_{i,j}^{PDF}$ を，文字トークンの場合と同様に，図形の描画命令種（[MOVE_TO]，[LINE_TO]，[CURVE_TO]）の埋め込みベクトルと，$\{f_i\}_{i=1}^{m_i}$ を列挙したベクトルを線形写像によって変換したものの和とする．ただし，線形写像のパラメータ行列は，描画命令種毎に異なるものを用意する．

次に，$\mathbf{h}_{i,j}^{PDF}$ に対して畳み込み演算を適用し，周辺の図形描画命令を考慮したベクトル表現 $\tilde{\mathbf{h}}_{i,j}^{PDF}$ を得る．ただし，畳み込み演算のウィンドウ幅を 5，出力チャネル数は d とし，零ベクトルでパディングする．そして，Max-Pooling により，ベクトル表現 $\mathbf{h}_i^{PDF}$ を得る．

$$\mathbf{h}_i^{PDF} = \max_{j=1}^{m} \tilde{\mathbf{h}}_{i,j}^{PDF}$$

表 3 の 6 番目のトークン t_6 に対する計算例を図 4 に示す．

最後に，周辺のトークンを考慮したベクトル表現を得るために，画像 Encoder の場合と同様に，$\{\mathbf{h}_i^{PDF}\}_{i=1}^{T}$ に対して双方向型 LSTM を適用して，d 次元素性ベクトル列 $\{\tilde{\mathbf{h}}_i^{PDF}\}_{i=1}^{T}$ を得る．

4.3 数式 XML 解析のための遷移システム

我々の提案する数式 XML 解析のための遷移型句構造解析手法は，入力のトークン列を保持するバッファと，これまでに決定された部分木を保持するスタックから成る．初期状態では，入力のトークン列は全てバッファに保持され

表 4　導出例．[x] は図形トークンを表す．

	直前の操作	σ	β
0	（初期状態）		s i n [–] θ [^] 2
1	SHIFT-COPY	s	i n [–] θ [^] 2
2	SHIFT-COPY	s i	n [–] θ [^] 2
3	SHIFT-COPY	s i n	[–] θ [^] 2
4	REDUCE-<mi‾>	s <mi‾>	[–] θ [^] 2
5	REDUCE-<mi>	<mi>	[–] θ [^] 2
6	SHIFT-REMOVE	<mi>	θ [^] 2
7	SHIFT-COPY	<mi> θ	[^] 2
8	UNARY-<mi>	<mi> <mi>	[^] 2
9	SHIFT-DRAW-^	<mi> <mi> ^	2
10	UNARY-<mo>	<mi> <mi> <mo>	2
11	REDUCE-<mover>	<mi> <mover>	2
12	SHIFT-COPY	<mi> <mover> 2	
13	UNARY-<mn>	<mi> <mover> <mn>	
14	REDUCE-<mfrac>	<mi> <mfrac>	
15	REDUCE-<math>	<math>	

表 5　アクションと状態遷移

アクション	現在の状態	次の状態	前提条件
SHIFT-COPY	$(\sigma, t\|\beta)$	$(\sigma\|t, \beta)$	$t \in \mathcal{C}$
SHIFT-DRAW-X	$(\sigma, t\|\beta)$	$(\sigma\|X, \beta)$	$t \in \mathcal{D}$
SHIFT-REMOVE	$(\sigma, t\|\beta)$	(σ, β)	—
REDUCE-X	$(\sigma\|s_1 s_0, \beta)$	$(\sigma\|X, \beta)$	—
UNARY-X	$(\sigma\|s_0, \beta)$	$(\sigma\|X, \beta)$	—

ており，SHIFT アクションによってバッファ先頭のトークンを 1 つ取り出し，ノードとしてスタックにプッシュする．また，REDUCE/UNARY アクションによって，スタック先頭のノードに対する部分木を決定する．

提案手法は，依存構造解析で用いられている shift-reduce 解析とよく似ているが，予測対象が依存構造ではなく句構造であること，入力トークンを削除する SHIFT-REMOVE というアクションが必要であること，入力系列が文字だけでなく図形情報も混在していることなどが異なる点として挙げられる．

図 1 に対する MathML の導出例を表 4 に示す．

表 5 に，遷移システムの各アクションと状態遷移を示す．遷移システムの状態は，部分木を要素に持つスタック σ と，入力トークンを要素に持つバッファ β の組 (σ, β) である．SHIFT-COPY アクションは，バッファの先頭のトークン t を取り出し，単一のノード t のみからなる部分木をスタックに追加する．ただし，t が文字トークンである場合（$t \in \mathcal{C}$）のみ適用可能である．SHIFT-DRAW-X アクションは，バッファの先頭のトークン t を取り出し，単一のノード X のみからなる部分木をスタックに追加する．ただし，t が図形トークンである場合（$t \in \mathcal{D}$）のみ適用可能である．SHIFT-REMOVE アクションは，バッファの先頭のトークン t を取り出し，スタックには何も追加しない．REDUCE-X アクションは，スタックの先頭 2 つの部分木 s_0，s_1 をポップし，ノード X が親，s_0 が右の子，s_1 が左の子である部分木を新たに作成し，スタックにプッシュする．UNARY-X アクションは，スタックの先頭の部分木 s_0 をポップし，ノード X が親，s_0 が唯一の子である部分木を新たに作成し，スタックにプッシュする．初期状態では，バッファが全ての入力トークンを保持し，スタックは空である．1 つずつアクションを適用していき，最終的に，バッファが空で，スタックの要素数が 1 かつスタック先頭の要素が<math>を親に持つ部分木になった状態で終了する．

この遷移システムでは，二分木のみ構築可能である．そのため，従来の句構造解析で行われているのと同様に，事前に MathML を right-branching 二分木に変換してから句構造解析を行い，解析後に後処理で元の多分木に戻す操作を行う．二分木化の過程で作られる中間ノードは，元の親ノード名に上線をつけて区別する（例：表 4 の <mi‾>）．

また，UNARY アクションは任意の回数繰り返すことが可能であり，解析が終了しない可能性がある．そこで，訓練データ中で複数回連続する UNARY アクションは 1 つの UNARY アクションに縮約し，アクション予測時に UNARY アクションを 2 回連続で予測できないような制約を加える．例えば，UNARY-mn，UNARY-msqrt という 2 つの連続する UNARY アクションは，UNARY-mn|msqrt という 1 つの UNARY アクションに縮約する．

4.4　フィードフォワードニューラルネットワークによるアクション予測

各時刻 t でのアクションは，スタックとバッファの状態から計算される特徴ベクトルを元に，フィードフォワードニューラルネットワーク（Feed Forward Neural Network; FFNN）によって予測する [1], [3].

スタックの特徴ベクトル

我々は Zhu ら [13] に従い，以下のスタック上の部分木に対する実数値ベクトルを結合して，スタックの特徴ベクトル $\mathbf{h}^{stack}$ とする．

$$\{s_3, s_2, s_1.left, s_1.unary, s_1.right, s_1,$$
$$s_0.left, s_0.unary, s_0.right, s_0\}$$

ただし，s_i はスタックの i 番目の部分木を表し，$s_i.left$ は s_i の左の子の部分木，$s_i.unary$ は s_i が子を 1 つだけ持つ場合の子の部分木，$s_i.right$ は s_i の右の子の部分木を表す．対応する部分木が存在しない場合は，零ベクトルを用いる．図 5 にスタックと $\mathbf{h}^{stack}$ の対応関係の例を示す．

スタック上の部分木 s の特徴ベクトルは，s の根ノードのタグの埋め込みベクトルと，入力トークン列中で s に対応するスパンの素性ベクトルを結合したものである．s の根ノードが葉である場合は，<LEAF>という特殊なタグを持つと考える．

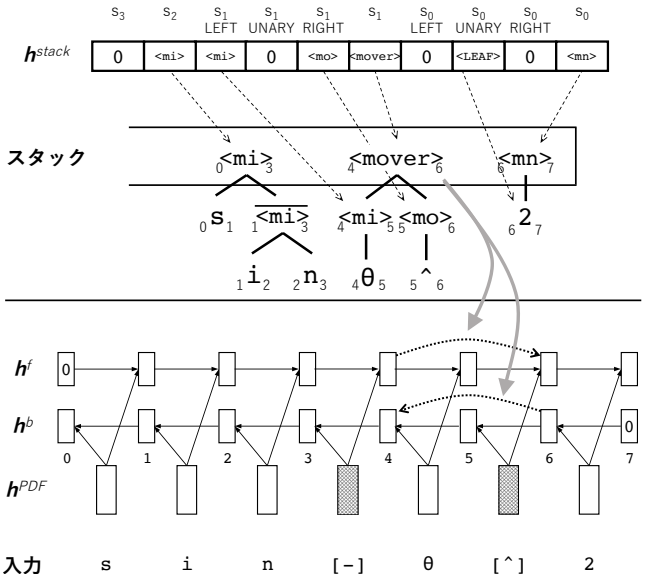

図 5 上：表 4 のステップ 13 におけるスタックの状態とスタックの素性 $\mathbf{h}^{stack}$，下：LSTM-Minus．灰色の矢印は $_4$<mover>$_6$ の LSTM-Minus によるスパン素性を指す．

また，句構造解析では，句に対応する入力スパンの特徴ベクトルを計算する必要がある．我々は，スパンの特徴ベクトルとして，Cross ら [4] による LSTM-Minus を使用した．これは，スパンの左端と右端の入力トークンに対応する隠れ状態ベクトルの差分を，そのスパンの特徴ベクトルとするモデルである．具体的な計算方法は以下の通りである．まず，PDF Encoder が計算する $\{\{\mathbf{h}_i^{PDF}\}_{i=1}^{T}\}$ に対して，順方向の LSTM による隠れ状態ベクトル $\{\tilde{\mathbf{h}}_0^f, ..., \tilde{\mathbf{h}}_T^f\}$，逆方向の LSTM による隠れ状態ベクトル $\{\tilde{\mathbf{h}}_0^b, ..., \tilde{\mathbf{h}}_T^b\}$ を求める．s が入力トークン列中の i 番目から j 番目までに対応しているとすると，スパンの素性は $\tilde{\mathbf{h}}_j^f - \tilde{\mathbf{h}}_i^f$ と $\tilde{\mathbf{h}}_i^b - \tilde{\mathbf{h}}_j^b$ を結合したものである．ただし，$\tilde{\mathbf{h}}_0^f$ と $\tilde{\mathbf{h}}_T^b$ を零ベクトルとする．

バッファの特徴ベクトル

バッファの特徴ベクトルを計算するために，我々は Zhu ら [13] に従い，バッファ上のトークンの実数値ベクトルを結合してバッファの特徴ベクトル $\mathbf{h}^{buffer}$ とする．すなわち，

$$\{q_0, q_1, q_2, q_3\}$$

ただし，q_i はバッファ上の i 番目のトークンを表す．また，q_i の特徴ベクトルには，PDF Encoder によって計算された素性ベクトル $\mathbf{h}_i^{PDF}$ を用いる．

4.4.1 FFNN の構成と学習

上記のスタックやバッファから計算される特徴ベクトルをフィードフォワードニューラルネットワーク（FFNN）へ入力し，各時刻で最適なアクションを決定する．具体的には，スタックの特徴ベクトル $\mathbf{h}^{stack}$ と，バッファの特徴ベクトル $\mathbf{h}^{buffer}$ を結合して，中間層が 1 層（d 次元）の FFNN に入力する．活性化関数には rectified linear unit (ReLU) を用いた．

モデルの学習は，各時刻 t におけるアクションの予測に対する交差エントロピー誤差関数を計算し，全ての時刻における和を最小化するようにパラメータの最適化を行う [1], [3].

5. 実験

2 節で構築した数式 XML 解析データセットを用いて，従来手法と提案手法の比較実験を行う．

5.1 比較手法

従来手法：ENCDEC-IMG-XML

数式 PDF を画像に変換し，3.1 節の画像 Encoder と 3.2 節の Decoder を組み合わせた Encoder-Decoder モデルにより，MathML のトークン系列を予測する．

提案手法 1：ENCDEC-PDF-XML

数式 PDF から文字・図形情報の系列を抽出し，4.2 節の PDF Encoder と 3.2 節の Decoder を組み合わせた Encoder-Decoder モデルにより，MathML のトークン系列を予測する．

提案手法 2：ENCDEC-PDF-ACTION

数式 PDF から文字・図形情報の系列を抽出し，4.2 節の PDF Encoder と 3.2 節の Decoder を組み合わせた Encoder-Decoder モデルにより，遷移システムのアクション系列を予測する．

提案手法 3：FFNN-PDF-ACTION

数式 PDF から文字・図形情報の系列を抽出し，**??**節の FFNN により遷移システムのアクション系列を予測する．

5.2 実験設定

デコード時の制約

デコード時は，各時刻においてスコアが最大である予測（MathML のトークン，遷移システムのアクション）を greedy に選択する．ただし，出力が木構造の制約を満たすように，その時点で適用可能な予測のみを行うように制約を与える．即ち，MathML のトークンを予測する場合は，閉じタグは直前に開いたタグに対応するもののみ予測できるようにする．遷移システムのアクションを予測する場合は，スタックとバッファの状態に基づき，その時点で適用可能なアクションのみを予測できるようにする．

Encoder-Decoder モデルは<EOS>タグを出力するまで生成を続けるが，<EOS>タグを生成しないため生成が停止しないような場合がある．そのような場合に対処するために，PDF から抽出したトークン数に応じて，出力トークン数に上限を設定する．MathML のトークンを予測する場合は入力トークン数の 7 倍（7 倍を超えるものが学習データになかったため）遷移システムのアクションを予測する場合は入力トークン数の 4 倍（理論上 4 倍を超えることはな

表 6　実験結果

手法	BLEU	句構造		
		適合率	再現率	F_1 値
ENCDEC-IMG-XML	81.73	0.790	0.804	0.797
ENCDEC-PDF-XML	81.05	0.812	0.841	0.826
ENCDEC-PDF-ACTION	82.53	0.790	0.838	0.814
FFNN-PDF-ACTION	**88.96**	**0.912**	**0.893**	**0.902**

いため）を出力トークン長の上限とする．

ハイパーパラメータ

各埋め込みベクトルの次元は 128 に，双方向 LSTM の隠れ状態ベクトルの次元は 256 に，FFNN の中間層の次元は 512 に設定した．LSTM は全て 1 層のものを用いる．

パラメータ最適化

数式 XML 解析データセットの訓練データを用いてパラメータ最適化を行った．長さが 200 トークンを超えるものは訓練・開発データから取り除いた．ただし，評価データには 200 トークンを超えるものは含まれている．最適化には Adam [7]（ハイパーパラメータは論文で提案されているものを使用）を用い，1 イテレーション毎に α パラメータを 0.99999 倍した．計算の効率化のために，入力トークン数が同じ事例を 8 つずつ集めてミニバッチを構成し，ミニバッチ単位で勾配の計算，パラメータの更新を行った．学習は 50 エポックまで行い，開発データでの尤度が最大となるエポックにおけるモデルを選択した．

5.3　評価手法

第一の評価指標として，MathML のトークン系列に対する BLEU スコア [9] を用いる．

第二の評価指標として，句構造解析で用いられる Evalb[*7]に基づく，句構造を考慮した評価指標を用いる．この評価指標では，句構造木中の各中間ノードに対して，中間ノードのラベル，ノード以下の部分木のの左端の葉ノード，右端の葉ノードの三つ組を抽出する．例えば，図 1 の MathML からは，次のような三つ組の集合が抽出できる：（<math>, s_1, 2_1), (<mi>, s_1, n_1), (<mfrac>, θ_1, 2_1), (<mover>, θ_1, ^$_1$), (<mi>, θ_1, θ_1), (<mo>, ^$_1$, ^$_1$), (<mn>, 2_1, 2_1) . ただし，入力中の同じ文字を区別するために，葉ノードの文字ごとに，左端から順番に番号を割り振る．正解と予測の句構造木それぞれから三つ組の集合を抽出し，それらの間の適合率，再現率，F_1 値で評価する．

5.4　実験結果

表 6 に，それぞれの手法における BLEU，句構造による適合率，再現率，F_1 値を示す．画像情報を用いるベースライン手法（ENCDEC-IMG-XML）に比べて，PDF から直接抽出した情報を用いる提案手法では，再現率と F_1 値が向上している．また，全ての評価指標において FFNN-PDF-ACTION の精度が，他の手法に比べて大幅に高いことがわかる．Encoder-Decoder を用いる手法では，生成が終了しない例が見られ，これが精度低下の大きな原因になっている．

[*7] http://nlp.cs.nyu.edu/evalb/

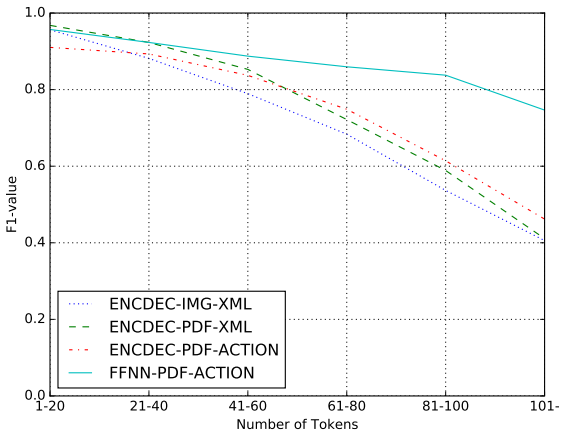

図 6　トークン数ごとの句構造 F_1 値

正解の MathML のトークン数ごとに句構造評価の F_1 値をプロットしたものを図 6 に示す．FF-PDF-ACTION は，特に長い数式に対して他の手法より高い精度で解析できていることがわかる．これは，Encoder-Decoder モデルと異なり，FFNN-PDF-ACTION はスタックとバッファから特徴ベクトルを計算することで，解析途中の部分木と入力位置を明示的に扱っているからだと考えられる．

6.　おわりに

本研究では，数式 PDF を MathML 形式に変換するという，数式 XML 解析タスクに取り組んだ．そのために，数式 PDF から直接抽出した文字・図形の情報を用いる， 遷移型句構造解析に基づく手法を提案した．実験により，画像情報のみを用いる従来手法に比べて，大幅に高い解析精度を達成することを示した．特に長い数式に対して，より頑健に解析できることが示された．

参考文献

[1] Andor, D., Alberti, C., Weiss, D., Severyn, A., Presta, A., Ganchev, K., Petrov, S. and Collins, M.: Globally Normalized Transition-Based Neural Networks, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany, Association for Computational Linguistics, pp. 2442–2452 (2016).

[2] Bahdanau, D., Cho, K. and Bengio, Y.: Neural machine translation by jointly learning to align and translate, *ICLR 2015* (2014).

[3] Chen, D. and Manning, C.: A Fast and Accurate Dependency Parser using Neural Networks, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, Association

for Computational Linguistics, pp. 740–750 (2014).
[4] Cross, J. and Huang, L.: Span-Based Constituency Parsing with a Structure-Label System and Provably Optimal Dynamic Oracles, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas, Association for Computational Linguistics, pp. 1–11 (2016).
[5] Deng, Y., Kanervisto, A. and Rush, A. M.: What You Get Is What You See: A Visual Markup Decompiler, *CoRR*, Vol. abs/1609.04938 (2016).
[6] Hochreiter, S. and Schmidhuber, J.: Long Short-Term Memory, *Neural Computation*, Vol. 9, No. 8, pp. 1735–1780 (online), DOI: 10.1162/neco.1997.9.8.1735 (1997).
[7] Kingma, D. P. and Ba, J.: Adam: A Method for Stochastic Optimization, *CoRR*, Vol. abs/1412.6980 (2014).
[8] Luong, T., Pham, H. and Manning, C. D.: Effective Approaches to Attention-based Neural Machine Translation, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, Association for Computational Linguistics, pp. 1412–1421 (2015).
[9] Papineni, K., Roukos, S., Ward, T. and Zhu, W.-J.: Bleu: a Method for Automatic Evaluation of Machine Translation, *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, Pennsylvania, USA, Association for Computational Linguistics, pp. 311–318 (online), DOI: 10.3115/1073083.1073135 (2002).
[10] Suzuki, M., Tamari, F., Fukuda, R., Uchida, S. and Kanahori, T.: INFTY: an integrated OCR system for mathematical documents, *Proceedings of the 2003 ACM symposium on Document engineering*, ACM, pp. 95–104 (2003).
[11] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R. and Bengio, Y.: Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)* (Blei, D. and Bach, F., eds.), JMLR Workshop and Conference Proceedings, pp. 2048–2057 (2015).
[12] Zanibbi, R., Aizawa, A., Kohlhase, M., Ounis, I. and Davila, K.: NTCIR-12 MathIR Task Overview.
[13] Zhu, M., Zhang, Y., Chen, W., Zhang, M. and Zhu, J.: Fast and Accurate Shift-Reduce Constituent Parsing, *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Sofia, Bulgaria, Association for Computational Linguistics, pp. 434–443 (2013).