自然言語処理データの単体複体による幾何的表現とそのホモロジー群

岩切 大葵[1]　佐藤 好久[2]

概要：近年のネット社会において，自然言語処理は検索，翻訳，誤情報分析など幅広い分野で活用され，不可欠な技術となっている．一方で，従来手法にはバイアスによる誤判断や大量のデータや計算資源を要する高コストといった課題が存在する．これらを踏まえ，本研究ではテキストデータの内部構造を位相幾何的に解析する手法を提案する．テキストデータに対して，幾何的モデリングを行い，単体複体を構成する．先行研究では，n-gram を用いた幾何的モデリングが提案されているが，本研究では，構文解析を用いて幾何的モデリングを行う．構成された単体複体のホモロジー群を用いた特徴抽出を通じて，テキストの構造を理解する有用性を検討する．

キーワード：情報数学-すべて，自然言語-すべて，テキスト処理

Geometric Representation of Natural Language Processing Data Using Simplicial Complexes and Their Homology Groups

Daiki Iwakiri[1]　Yoshihisa Sato[2]

Abstract: In today's networked society, natural language processing (NLP) has become an indispensable technology, widely applied in areas such as search engines, translation, and misinformation analysis. However, conventional methods face challenges including biased misjudgments and the high cost of requiring large amounts of data and computational resources. In light of these issues, this study proposes a method for topologically analyzing the internal structure of text data. We perform geometric modeling on text data to construct simplicial complexes. While prior research has proposed geometric modeling using n-grams, this study employs syntactic analysis for geometric modeling. We examine the usefulness of understanding text structure through feature extraction using the homology groups of the constructed simplicial complexes.

Keywords: Information Mathematics - All, Natural Language - All, Text Processing

1. はじめに

自然言語処理技術（NLP）は，インターネットの普及とともに急速に発展し，検索エンジン，翻訳，感情分析など，様々な応用分野で重要な役割を果たしている．従来の手法では，単語や文を高次元のベクトルとして捉え，機械学習モデルに適用するアプローチが主流である．例えば，Word2Vec は単語の共起情報を用いて分散表現を学習し，BERT は文脈情報を考慮して単語ベクトルを生成する．しかし，これらの手法は言語データの高次元構造を捉えるには不十分であり，ベクトル空間における単なる線形関係に基づく解析にとどまっている．また，これらの手法にはデータの冗長性や，学習データのバイアスの強い影響による不正確な情報や偏った情報の生成などといった問題がある．実際に，2017 年に Google の検索アルゴリズムが「オバマ元大統領がクーデターを計画している．」といった誤情報を上位表示してしまい，誤情報が信頼性のある情報よ

[1] 九州工業大学大学院 情報工学府 情報創成工学専攻
Graduate School of Computer Science and Systems Engineering, Kyushu Institute of Technology
[2] 九州工業大学大学院 情報工学研究院
Kyushu Institute of Technology

り優先される問題が発生した．他にも，同年に Facebook の AI 翻訳がアラビア語の「おはよう」問フレーズを「攻撃しろ」と誤訳し，無実のパレスチナ人男性がテロリストとして誤認逮捕されるという事件が発生した．こういった問題のために，テキストデータの内部構造をより深く解析したり，バイアスや誤情報が言語データの内部でどのような構造を持つかを解析する新たな視点が必要である．

そこで，高次の関係性やトポロジー的な構造を捉えるために有効な数学的手法である単体複体に着目する．本研究の目的は，自然言語処理データを単体複体として幾何的に表現し，さらにそのホモロジー群を用いて言語の構造的特徴を解析する新たな手法の有用性について確かめるものである．この手法により，従来の NLP モデルではとらえきれなかった言語データのトポロジカルな特徴を解析し，バイアスの可視化や誤情報検出，データの次元削減と効率化などに応用することを目指す．

ここで，J.D.Boissonnat による Building Efficient and Compact Data Structures for Simplicial Complexes[1] を参考にして，構文解析の二分木から単体複体を作成する着想を得た．また，Stephen Fitz による The Shape Of Words-TOPOLOGICAL STRUCTURES IN NATURAL LANGUAGE DATA[3] を参考にして，n-gram 手法で単体複体を作成する着想を得た．

2. 準備

2.1 幾何的準備

定義 2.1 (単体) 十分大きな $N \in \mathbb{N}$ に対して，$a_0, a_1, a_2, \ldots, a_m \in \mathbb{R}^N$ を一般の位置にある $m+1$ 個の点とし，

$$\sigma^m := \left\{ \sum_{i=0}^{m} \lambda_i a_i \;\middle|\; \sum_{i=0}^{m} \lambda_i = 1, \lambda_0, \lambda_1, \ldots, \lambda_m \geq 0 \right\}$$

を m **次元単体**という．各 a_i を単体 σ^m の**頂点**といい，単体の頂点を明記したいときには，

$$\sigma^m = |a_0, a_1, \cdots, a_m|$$

と表すこととする．
m 次元単体 $\sigma^m = |a_0, a_1, \cdots, a_m|$ の $m+1$ 個の頂点のうちのいくつか，例えば，$a_{i_0}, a_{i_1}, \cdots, a_{i_k}$ を頂点とする k 次元単体 $\tau^k = |a_{i_0} a_{i_1} \cdots a_{i_k}|$ を σ^m の k **次元面**といい，

$$\tau^k \prec \sigma^m$$

と書く．m 次元単体 σ の $(m-1)$ 次元単体は σ の「境界」である．また，0 次元面は頂点であり，1 次元面は辺である．例として，0-単体は点，1-単体は線分，2-単体は三角形，3-単体は中身が詰まった三角錐である．

定義 2.2 (単体複体) 高々加算個の単体からなる集合 $K = \{\sigma, \tau, \ldots, \}$ が次の条件 (1)(2) を満たすとき，K は**単体複体**（簡単に，**複体**）という．

(1) $\sigma \in K, \tau \prec \sigma \implies \tau \prec K$ （K の任意の単体について，その任意の面はまた K に属する．）

(2) $\sigma, \tau \in K, \sigma \cap \tau \neq \emptyset \implies \sigma \cap \tau \prec \sigma, \tau$ （K の 2 つの単体の交わりは，それぞれの単体の面である．）

2.2 ホモロジー群

単体同士の「和」,「差」,「定数倍」を考えたい．そのために単体複体内の各単体に任意の向きを割り当てる必要がある．単体の向きは単体の頂点への順番の付け方のことであり，その順番の付け方が置換としての偶奇が同じものである場合，単体の向きは同じであると考える．向きを持つ単体は，各括弧内に頂点を並べた順序で表記する．

$$\sigma = \langle a_0 a_1 \ldots a_k \rangle$$

また，任意の向きを持つ単体 σ に対し，逆向きの同じ単体を $-\sigma$ と表記する．

定義 2.3 (鎖群) 複体 K の各 q 次元単体に対して 1 つの向きを定め，これらの向き付けられた q 次元単体によって生成される自由 $\mathbb{Z}$ 加群を

$$C_q(K) := \left\{ \sum_{i=1}^{s} m_i \langle \sigma_i^q \rangle \;\middle|\; m_1, m_2, \ldots, m_s \in \mathbb{Z} \right\}$$

と書いて，K の q **次元鎖群**という．また，便宜的に，

$$C_q(K) = 0 \quad (q > \dim K), \quad C_{-1}(K) = 0$$

とする．

定義 2.4 (境界作用素) 鎖群の間の準同型写像 $\partial_q : C_q(K) \to C_{q-1}(K)$ を各生成元 $\langle \sigma \rangle = \langle a_0 a_1 \cdots a_q \rangle$ に対して，

$$\partial_q(\langle \sigma \rangle) := \sum_{i=0}^{q} (-1)^i \langle a_0 a_1 \cdots \hat{a}_i \cdots a_q \rangle$$

と定める．任意の元に対しては，自由 $\mathbb{Z}$ 加群の準同型として自然に拡張して定義する．すなわち，

$$\partial_q \left(\sum_{j=1}^{s} m_j \langle \sigma_j^q \rangle \right) = \sum_{j=1}^{s} m_j \partial_q(\langle \sigma_j^q \rangle)$$

と定める．この準同型写像 $\partial_q : C_q(K) \to C_{q-1}(K)$ を複体 K の**境界準同型**または**境界作用素**という．

定義 2.5 (鎖複体) 複体 K に対して，$\{(C_q(K), \partial_q) | q \in \mathbb{Z}\}$ を複体 K の**鎖複体**という．鎖複体から定められる $C_q(K)$ の重要な部分群を 2 つ定義する：

$$B_q(K) := \mathrm{Im}\ \partial_{q+1}, \quad Z_q(K) := \mathrm{Ker}\ \partial_q$$

$B_q(K)$ を複体 K の q **次元境界輪体群**，$Z_q(K)$ を複体 K の q **次元輪体群**という．特に，$Z_q(K)$ の元を q **次元サイクル**という．

定義 2.6 (ホモロジー群) 複体 K に対して，商群（剰余加群）

$$H_q(K) := Z_q(K)/B_q(K)$$

を複体 K の q **次元ホモロジー群**という．

定義 2.7 (ベッチ数) 一般に，有限複体 K に対して，ホモロジー群 $H_q(K)$ は有限生成アーベル群になることが知られている．したがって，有限生成アーベル群に関する基本定理を用いると，

$$H_q(K) \cong \underbrace{\mathbb{Z} \oplus \mathbb{Z} \oplus \cdots \oplus \mathbb{Z}}_{r} \oplus \mathbb{Z}_{n_1} \oplus \cdots \oplus \mathbb{Z}_{n_k}$$

であることが分かる．ただし，$\mathbb{Z}_n$ は位数 n の巡回群である．この分解に現れる無限巡回群 $\mathbb{Z}$ の個数を $b_q(K) = r$ と書き，複体 K の q **次元ベッチ数**という．

2.3 構文解析

構文解析とは，文の構文的な構造を決定することであり，句構造文法が使われることが多い．文法による構文木は一般に複数あるが，構文木の違いというのは解釈の違いということである．構文解析の目的は，句構造文法の規則を使って，文を生成できる構文木をすべて見つけ出すことである．

文法規則には 2 種類ある：

- 句構造規則 . . . 非終端記号と前終端記号からなる規則
- 辞書規則 . . . 前終端記号 → 単語　という規則

（前終端記号とは，単語を生成する前のシンボルであり，品詞に相当する．）

例として，I broke a desk with a drawer という文を構文解析する．

句構造規則として，

S → NP VP,　VP → v,　NP → pron,　VP → v NP,　NP → det n,　VP → VP PP,　NP → NP PP,　PP → prep NP

辞書規則として，

pron → I,　det → a,　v → broke,　prep → with,　n → desk, drawer

を使用する．

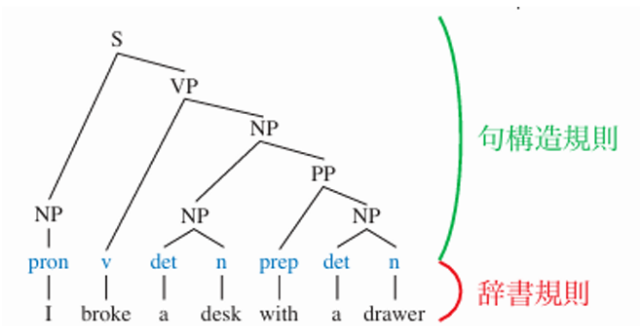

図 1 解析木

構文解析の入力は，単語列を入力とすると，膨大な数の辞書規則が必要となるので，実際には品詞列を入力とする場合が多い．単語と品詞の対応付けは形態素解析で行い，前終端記号を終端記号として扱い，句構造規則のみ使う．

3. Simplex Tree

3.1 Simplex Tree

K を単体複体とする．ここで，K の頂点は 1 から n までの番号が付けられており，その順序に従うものとする．したがって，各単体は $1, \ldots, n$ というアルファベットセット上の単語として表現できる．

この単体の集合をコンパクトに表現するため，単体を**ツリー構造**に格納する．このデータ構造は **ST**(**K**) または単に **ST** と表現される．

このツリーは，単体の単語を格納するデジタル検索木のような構造を持つ．

- ルートの深さは 0
- ノードの深さは，そのノードが表す単体の次元 +1 に等しい

ST は以下の手順で構築される．

(1) 空のツリーから開始
(2) 単体の単語をツリーに挿入
(3) ルートから開始し，既に存在する最長の部分パスをたどる．その後，ツリー内に存在しない部分のノードを新規に追加

ST は冗長性があるので，これを解消するため**圧縮 Simplex Tree**（C(ST)）を導入する．

3.2 圧縮 Simplex Tree（C(ST)）

ST の構造には，同じ部分木が複数回登場するという冗長性がある．それらの重複する部分木を 1 つだけ保持することでツリーを圧縮する．

- 同じ部分木が異なるノードに存在する場合，それらを 1 つに統合する
- 統合後のノードに複数の親ができる
- 圧縮後のツリーは ST のノードと 1 対 1 の対応を持たない
- ルートからのパスは ST と C(ST) で一意に対応する

圧縮後も，ST の基本操作はそのまま実行可能である．また，親ノードが一意でないため，上向きの探索には追加のポインタを利用する．

4. 手法

4.1 構文解析手法 (提案手法)

構文解析の結果を単体複体として表すために必要な考えとして，単体複体から ST を構築する手順がある．本研究では逆に構文解析の結果から作成した二分木から単体複体を作成した．　二分木から単体複体を作成する手法は以下

の 3 ステップで行う．

Once upon a time, there was an old residence surrounded by moats. を例文として，各ステップの具体例を示す．

ステップ 1：根から子をたどっていき，子がなくなったら単体として記録

構文木の根から子ノードを順にたどり，子がなくなった（葉ノードに到達した）時点で，たどってきた経路上のノードを 1 つの単体として記録する．これにより得られる単体は (was, Once), (was, upon, time, a), (was, [,]), (was, there), (was, residence, an), (was, residence, old), (was, residence, surrounded, by, moats), (was, [.]) 等である．以下はステップ 1 で作成される単体の一例である．

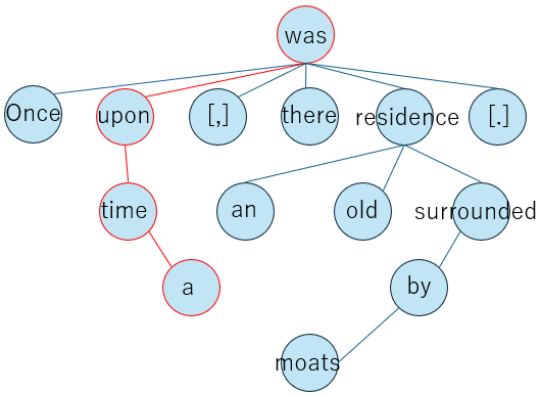

図 2 ステップ 1 により作成される単体の例

ステップ 2：同じ親を持つ子同士を単体として記録

同じ親ノードを持つ子ノード同士を 1 つの単体として記録する．これにより得られる単体は (Once, upon, [,], there, residence, [.]), (an, old, surrounded) 等である．以下はステップ 2 で作成される単体の一例である．

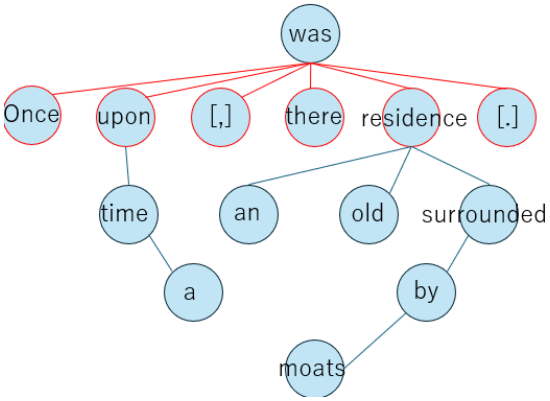

図 3 ステップ 2 により作成される単体の例

ステップ 3：各単体から部分単体を列挙，重複除去して単体複体を作成

ステップ 1,2 で得られた各単体から部分単体をすべて列挙し，重複を除去して単体複体を作成する．例えば，(upon, time, a) の部分単体は (upon), (time), (a), (upon, time), (upon, a), (time, a), (upon, time, a) である．ただし，(an, surrounded, by) のように同じ親を持たないノードの組は部分単体ではない．以下は，赤線は作成される部分の例であり，黄色戦は作成されな単体の例である．

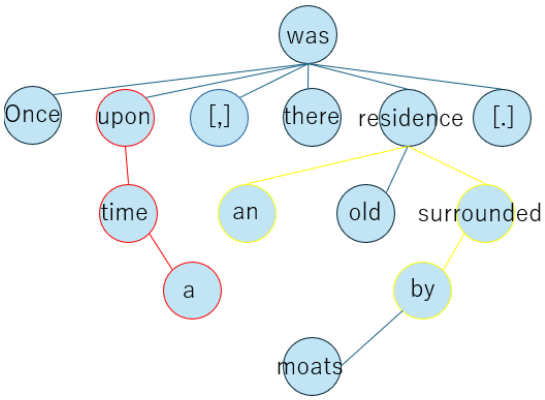

図 4 ステップ 3 により作成される単体の例

4.2 n-gram 手法 (先行研究)

n-gram とはテキストデータを連続する n 個の単語の単位に分割する手法である．この技術は自然言語処理の基本的なアプローチの一つであり，テキストの統計的な特徴を捉えるために利用される．**ステップ 1：n の値を決め，連続した n 語を 1 語ずつずらしていきながら抽出ステップ 2：抽出した n 語を頂点として単体を作成ステップ 3：各単体から部分単体を列挙，重複除去して単体複体を作成**

5. 実験

5.1 目的

本研究の目的は，物語や小説に対して提案手法である構文解析方法と先行研究で提案されている n-gram を用いた方法でパラグラフごとに単体複体を構成し，ベッチ数を求め，折れ線グラフを表示して異なる手法により物語の構造的特徴を明らかにすることができるか検証することである．

本研究により，従来の自然言語処理とは異なる視点で物語の構造的特徴を数学的に解析できることを期待する．

5.2 使用したテキストデータ

以下は使用したテキストデータの 1 パラグラフ目である．

(1) **赤ずきん**

Once upon a time there was a sweet little girl. Everyone who saw her liked her, but most of all her grandmother, who did not know what to give the child next. Once she gave her a little cap made of red velvet. Because it suited her so well, and she wanted to wear

it all the time, she came to be known as Little Red Riding Hood.

5.3 手順

今回の研究の手順を以下に示す.

(1) パラグラフごとに複数のテキストを構文解析
(2) 構文解析した結果から二分木作成
(3) 二分木から単体複体作成
(4) 単体の部分集合をすべて生成
(5) 複数のテキストにより得られた単体複体を合成した一つのパラグラフの単体複体作成
(6) 単体複体を構成する単語をラベル付け
(7) それぞれの単体複体に対してホモロジー群，ベッチ数を計算．ホモロジー群は Python を利用して計算した [2]
(8) n の値を決定
(9) n 語を 1 語ずつずらしながら抽出
(10)抽出した n 語を頂点として単体複体作成
(11)(4) (7) を実行
(12)パラグラフを時間経過とした手法ごとの結果を比較する折れ線グラフを表示，比較

5.4 実験結果

以下にそれぞれの手法により得られたベッチ数を示す.

```
ParagraphID 1 (over Q): β_0=1, β_1=11, β_2=0, β_3=0
ParagraphID 2 (over Q): β_0=1, β_1=6, β_2=0, β_3=0
ParagraphID 3 (over Q): β_0=1, β_1=21, β_2=0, β_3=0
ParagraphID 4 (over Q): β_0=2, β_1=6, β_2=0, β_3=0
ParagraphID 5 (over Q): β_0=1, β_1=11, β_2=0, β_3=0
ParagraphID 6 (over Q): β_0=3, β_1=3, β_2=0, β_3=0
ParagraphID 7 (over Q): β_0=1, β_1=10, β_2=0, β_3=0
ParagraphID 8 (over Q): β_0=1, β_1=1, β_2=0, β_3=0
ParagraphID 9 (over Q): β_0=1, β_1=5, β_2=0, β_3=0
ParagraphID 10 (over Q): β_0=1, β_1=13, β_2=0, β_3=0
ParagraphID 11 (over Q): β_0=1, β_1=2, β_2=0, β_3=0
ParagraphID 12 (over Q): β_0=1, β_1=44, β_2=0, β_3=0
```

図 5 4-gram 手法により得られたベッチ数

n-gram では，単体複体の最大次元を設定しているので，求められるベッチ数の最大次元は全てのパラグラフで等しくなっている．4-gram の結果で，0 次元ベッチ数が 1 でないパラグラフが 4 つ目と 6 つ目に出現しているが，これはその文に出現した単語がパラグラフ内の他の文に出現していないことを表す.

```
ParagraphID 1 (over Q): β_0=1, β_1=44, β_2=0, β_3=0, β_4=0, β_5=0, β_6=0
ParagraphID 2 (over Q): β_0=1, β_1=51, β_2=0, β_3=0, β_4=0, β_5=0, β_6=0
ParagraphID 3 (over Q): β_0=1, β_1=93, β_2=0, β_3=0, β_4=0, β_5=0, β_6=0
ParagraphID 4 (over Q): β_0=1, β_1=65, β_2=0, β_3=0, β_4=0, β_5=0, β_6=0
ParagraphID 5 (over Q): β_0=1, β_1=46, β_2=0, β_3=0, β_4=0, β_5=0, β_6=0
ParagraphID 6 (over Q): β_0=1, β_1=64, β_2=0, β_3=0, β_4=0, β_5=0
ParagraphID 7 (over Q): β_0=1, β_1=66, β_2=0, β_3=0, β_4=0, β_5=0, β_6=0
ParagraphID 8 (over Q): β_0=1, β_1=24, β_2=0, β_3=0, β_4=0, β_5=0
ParagraphID 9 (over Q): β_0=1, β_1=32, β_2=0, β_3=0, β_4=0, β_5=0
ParagraphID 10 (over Q): β_0=1, β_1=104, β_2=0, β_3=0, β_4=0, β_5=0, β_6=0
ParagraphID 11 (over Q): β_0=1, β_1=38, β_2=0, β_3=0, β_4=0, β_5=0, β_6=0, β_7=0, β_8=0
ParagraphID 12 (over Q): β_0=1, β_1=201, β_2=0, β_3=0, β_4=0, β_5=0, β_6=0, β_7=0, β_8=0
```

図 6 構文解析手法により得られたベッチ数

構文解析手法で，パラグラフごとに求めているベッチ数の次元が異なるのは，単体複体の最大次元が異なるからである．0 次元ベッチ数がすべて 1 なのは，n-gram 手法とは異なり構文解析手法はピリオドも 1 つの語句として処理しているからである．パラグラフ内の他の文に出現しない単語を持つ文があっても，ピリオドが出現していればそのパラグラフの連結成分は 1 となる.

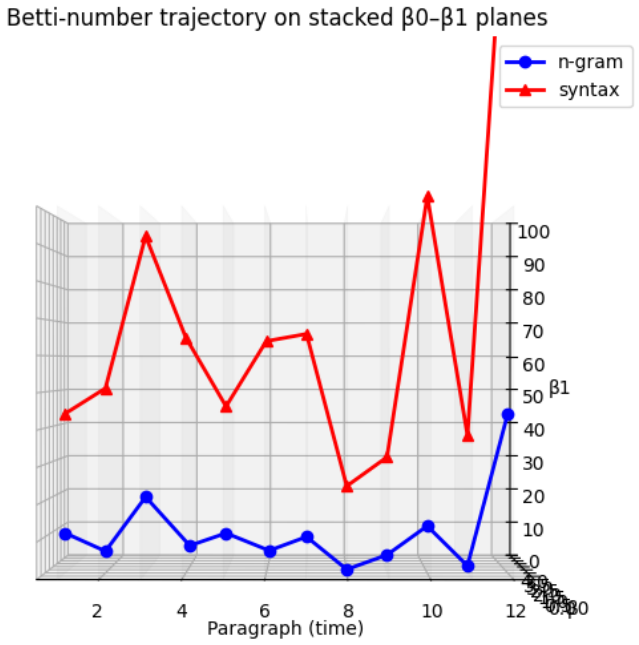

図 7 4-gram 手法と構文解析手法から得られたベッチ数の折れ線グラフ（平面）

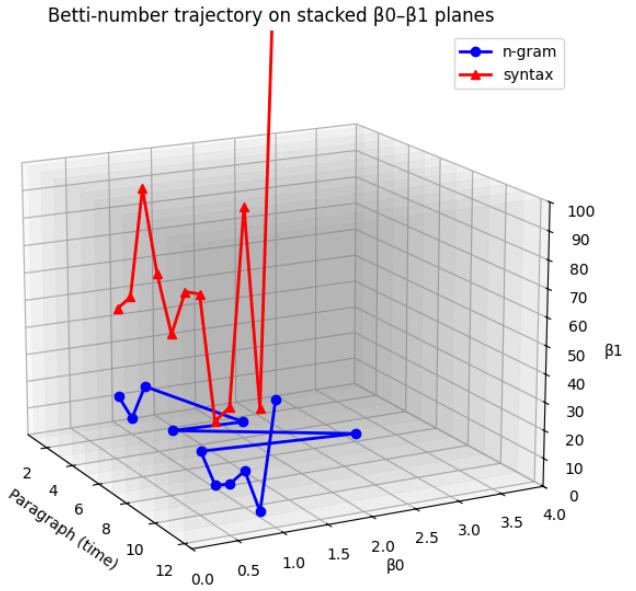

図 8 4-gram 手法と構文解析手法から得られたベッチ数の折れ線グラフ（立体）

赤色の線が構文解析手法で，青色の線が n-gram 手法の

線である．0 次元ベッチ数に関しては 4-gram 手法により得られた結果からわかるように，4 つ目と 6 つ目のパラグラフだけ 2 と 3 になった．1 次元ベッチ数に関しては，ベッチ数の大きさは異なるが似たグラフの形となった．

```
TextID 1 (over Q): β_0=1, β_1=6, β_2=1, β_3=0, β_4=0
TextID 2 (over Q): β_0=1, β_1=5, β_2=0, β_3=0, β_4=0
TextID 3 (over Q): β_0=1, β_1=17, β_2=1, β_3=0, β_4=0
TextID 4 (over Q): β_0=2, β_1=4, β_2=0, β_3=0, β_4=0
TextID 5 (over Q): β_0=1, β_1=8, β_2=0, β_3=0, β_4=0
TextID 6 (over Q): β_0=3, β_1=3, β_2=0, β_3=0, β_4=0
TextID 7 (over Q): β_0=1, β_1=8, β_2=0, β_3=0, β_4=0
TextID 8 (over Q): β_0=1, β_1=1, β_2=0, β_3=0, β_4=0
TextID 9 (over Q): β_0=1, β_1=4, β_2=0, β_3=0, β_4=0
TextID 10 (over Q): β_0=1, β_1=10, β_2=0, β_3=0, β_4=0
TextID 11 (over Q): β_0=1, β_1=2, β_2=0, β_3=0, β_4=0
TextID 12 (over Q): β_0=1, β_1=26, β_2=3, β_3=0, β_4=0
```

図 9 5-gram 手法により得られたベッチ数

5-gram 手法でベッチ数を求めると，構文解析手法で出現したことがなかった 2 次元ベッチ数が 1,3,12 パラグラフで出現した．

6. 結論，考察

・0 次元ベッチ数

少しの違いは出たものの，ほとんどのパラグラフで 1 なので重要ではないと考える．

・1 次元ベッチ数

テキスト数が他より多いパラグラフは 1 次元ベッチ数が多くなっていた．8 つ目のパラグラフはテキスト数は平均だが，同じような文章が数回繰り返されていただけなので 1 次元ベッチ数が 1 番少なくなったと考える．つまり，1 次元ベッチ数は単に文章量ではなく，情報量と比例していると考えられるので，文を理解するうえで重要だと考えられる．

・手法の比較

1 次元ベッチ数のグラフの形を比較してみると，構文解析手法の方が値が大きい．また，ほとんど形は同じだが，5,6 パラグラフだけ形が異なる．

さらに，今回は n=4 として実験をし，どちらの手法でも 2 次元以上のベッチ数は出てこなかったが，n=5 としたとき，n-gram 手法により求められたベッチ数の中で構文解析手法で出てきたことがなかった 2 次元ベッチ数が出てきた．

以上のことより，1 文の場合は n-gram 手法よりも構文解析手法の方が，最大次元が大きいため有効である．複数の文の場合は，どちらも適用可能である．表 1 に手法の適用範囲を示す．

表 1 手法の適用範囲の比較

	1 文	複数の文
n-gram	×	○
構文解析	○	○

参考文献

[1] Jean-Daniel Boissonnat, Karthik C.S., Sebastien Tavenas: Building Efficient and Compact Data Structures for Simplicial Complexes, Leibniz International Proceedings in Informatics (2015).
[2] Qiita: ホモロジー群を計算しよう (online), 入手先 ⟨https://qiita.com/hiro949/items/9c793415313be6312154⟩ (2022).
[3] Stephen Fitz: The Shape Of Words – TOPOLOGICAL STRUCTURES IN NATURAL LANGUAGE DATA, ICML 2022 Workshop on Topology, Algebra, and Geometry in Machine Learning (2022).