# 修　　士　　論　　文

# 学術論文の章構造に基づく<br>階層的ニューラル要約モデル

**A Hierarchical Neural Summarizer<br>for Academic Papers based on Chapter Structure**


指導教員　　鶴岡 慶雅 准教授

東京大学大学院工学系研究科<br>電気系工学専攻

氏 名　　**37-166450 衣川 和尭**


提 出 日　　平成 30 年 2 月 1 日

## 概 要

自然言語処理において文章の自動要約技術は最も重要な課題の一つである. このタスクは研究としても長い歴史を持つが, インターネットを通じてテキストデータが増え続けている現状を鑑みても今後一層重要になっていくと考えられる. 自動要約の技術にはいくつかの種類があるが, そのうちの一つに入力文書中の重要な文をいくつか抜粋してつなぎ合わせることで要約を作成する抽出型要約がある. 抽出型要約は比較的低い計算コストで, ある程度文法・意味的に正しい要約を作れるのが特徴である. 近年では特に深層学習の登場によって, 他の言語処理タスク同様, 抽出型要約の技術も盛んに研究されている. 言語処理において深層学習モデルの基礎となっているのがリカレントニューラルネットワークと呼ばれる系列情報の処理に適したニューラルネットワークである. テキストデータは単語の系列, 文の系列として表現できるため, リカレントニューラルネットワークは自然言語処理の多くの研究でよく用いられている. しかし一方で, 入力文書のサイズが大きくなると読み込ませる系列が必然的に長くなり, リカレントニューラルネットワークで処理するのが難しくなることが知られている. 近年の機械学習ベースの要約の研究の多くは比較的短いテキストをデータセットとして用いており, 学術論文のようなサイズの大きな文書の要約は依然として大きな課題である. また, 単純に文書が大きいと文の候補が多すぎて適切に抽出するのが困難になるという問題もある.

本研究では, 学術論文を対象とした抽出型要約のニューラルネットワークモデルを構築することを目指す. 要約に限らず長いテキストに対してどのようにモデルを組むかということは盛んに研究されているが, その対策の一つとして文書構造に則ってリカレントニューラルネットワークを階層的に構築することが挙げられる. 論文は一般的にセクション, パラグラフ, 文, 単語の順に木構造化できることに着目し, 本研究は論文の文書構造に沿った階層的なネットワーク構造を提案する. 次に, 文の数が膨大なために適切な文集合を抽出することが難しい問題についても, 木構造に沿って重要度のスコアを階層的に計算する手法を提案する. 一般に, 抽出型要約を教師有り学習に基づいて行う場合は, あらかじめ本文中の全ての文に対して正例・負例のラベルを貼ってそれを学習データとしているが, 本研究ではさらにパラグラフとセクションにもラベルを設けスコアを計算させる. そして, 木構造の各ノードでのスコアの大小に応じてより正しい文を選択できるようモデルをサポートする. この手法には同時に本筋とは関連性の薄い文を候補から外すことも期待できる.

生命医学系の論文をデータベースから収集して, それらを抽出型要約のためのデータセットとして評価実験を行った. 既存手法を勝る精度を達成するだけでなく, より本質と関連性の薄い文を除去するという提案の有効性を確認した.

# 目 次

# 図 目 次

# 表 目 次

# 第1章　序論

## 1.1　背景

情報化社会の現代において, 我々の身の回りのテキストデータは日ごとに増えている. 新聞や書籍, テレビのニュースといった従来までのメディア媒体に加え, 近年は SNS やインターネットニュース, 電子掲示板も広く世間に浸透するようになり, 自分の興味のあるものを全て読むことは不可能に近い. そうなると, 情報が増えれば増えるほど大事なところだけを知りたいというユーザー側の需要も高まってくるため, 必然的にそれを実現する技術の必要性も高まってくるであろう.

計算機に人間の言語を理解させることを目指す自然言語処理という学問において, 文書を自動的に要約させる技術は古くから研究されてきたが, こうした時代背景も受けて自動要約技術の研究はますます重要になっている. 自動要約とは文書を計算機に入力し, 重要な情報を保持しつつより短いサイズの文書を出力するタスクであるといえる. 自然言語処理における要約の研究の歴史は長く, 50 年以上前から研究されてきた [1]. 自動要約技術は抽出型要約と生成型要約の二つの種類に大別される. 前者は入力文章の中から重要だと思われる文や単語を抽出してつなげる要約である. それに対し後者は, 人間と同じように文章を意味的に再構成したり元の文章には含まれない語彙を使って行う要約である. 図 1.1 に抽出型要約と生成型要約の例を挙げる. 入力文書は四つの文で構成され, 簡単にいうと「車に乗って仕事場に出かけた」といった内容である. 抽出型要約では間の二番目および三番目の文は車に乗っているときの様子を表しているに過ぎないため除外され, 最初と最後の文を抜き出してつなげている. 一方生成型要約では, 人間が要約を行う時のように入力文の文意を保ちつつ, 適切に語彙を言い換えたり文をつなげたりしている. 例えば “drove” や “work” などは入力文書には出てこない.

生成型要約と抽出型要約にはそれぞれ以下のような長所と短所がある.

**文章の表現能力**

抽出型要約では一文一文の中身は固定されたまま出力されるのに対し, 生成型要約では, パラフレーズなどより自由で人間らしい表現が可能となる. また, 抽出型要約では, 抽出してきた文をつなげた場合に文脈的に不自然となる可能性がある.

**文法・意味的正しさ**

抽出型要約では, すでに完成された文をそのまま出力することになるので, 少なくとも文単位で見たら出力要約は文法・意味的に正しいことが保証されている. 一方で生成型要約では基本的には一単語ずつ予測していくことになるので, うまく生成できない可能性がある. 具体的には同じ単語やあるいは同じ文が繰り返し出力されてしまう. たとえば本来には “ This is a desk . That is a chair . ” と出力されるべきところが, “ This is a desk desk desk (...) ” となったり,

**Original Text:**
I walked out of the house and got in the car.
It started rough, and I let it idle for a few minutes.
I was late enough so that most of the traffic had cleared off of the freeway.
I pulled into the parking garage under my building at 8:45.

---

**Extractive summary:**
I walked out of the house and got in the car.
I pulled into the parking garage under my building at 8:45.

---

**Abstractive summary:**
I drove to work.

図 1.1: 抽出型要約および生成型要約の例 [1]. 赤い単語は元々の入力文書から抽出してきた単語であることを表し, 青い単語は元々の入力文書には含まれていない単語であることを表す.

> “ This is a desk . This is a desk . (...) ” となってしまうという問題である. 特に入力文が長いほどこれらの問題が起きやすいことが知られている [3]. また, ある述語に関する主語と述語を取り違えて出力してしまい, 意味的に全く異なる要約になってしまうといった問題も報告されている [4].

**計算コスト**

現状の生成型要約ではニューラルネットワークを用いた機械学習ベースの手法が主流となっている. 現在よく用いられているフレームワークでは数万のオーダーの単語候補から一語ずつ正しい単語を予測することになり, これはつまり数万クラス分類を出力単語数ぶん行うことと等価であるため, 基本的に計算コストが大きい. 一方で抽出型要約を機械学習で行う場合は, 各文に対して抽出するかしないかの二値分類となるため計算コストは比較的小さくなる [5].

歴史的には生成型要約のタスクは技術的に困難であると長い間考えられていたため, 多くの研究が抽出型要約を対象としてきた. しかし深層学習の成功により生成型要約も高い精度が出せるようになり, 近年ではむしろ生成型要約の方が盛んに研究されるようになっている [3, 6, 7]. とはいえ抽出型要約も先に述べたように, その学習コストが大きくないことや出力要約の文法や意味的正しさが保証されていることなどメリットは大きい. 事実, 機械学習ベースの抽出型要約の研究は従来的には特徴量工学に基づくアプローチが主であったが, ニューラルネットワークを用いて単語や文を密な高次元ベクトルに落とし込む手法が盛んに研究されるようになって, ベクトル表現された文を対象に最適化手法を用いる研究がでており [8], さらに, Cheng と Lapata [9] がよりディープにリカレント・ニューラルネットワークを活用したモデルを提案してから, 後続の研究がそれに続いている [5, 10, 11] など, 活発に研究されている.

要約の実験で用いるデータセットとしてはニュース記事 [9] や, レビューサイトのコメント [8],

ブログのコメント [12], アルバム [13] など様々なものが存在するが, その一つとして学術論文がある [14, 15, 16]. 要約データセットとしての論文の特徴としては, アブストラクトがあらかじめ付随しており, それは信頼性が高く質の良い正解要約とみなせるため, 要約のデータセットとして適していることや, 論文データベースから大量にダウンロード可能なため, 機械学習ベースの手法を適用しやすいことがあげられる. ここで, 論文の要約には大きく分けて二通りのタスクがある. 一つは, あるターゲット論文に対して, 複数の後続の論文中でそれについて述べている言説を集めるタスクである [15]. これはある研究に対して様々な側面から見たときの知見を回収するのが目的のタスクで, カテゴリーとしては複数文書の抽出型要約であるといえる. もう一つは論文の本文から一つの要約を作成するタスク [14, 16] である. 論文にはあらかじめアブストラクトという要約が付いているものの, 本文は特定の名前が付与されたセクション内ごとに文を区切ることで構造化されているのに対し, アブストラクトはそのように構造化されていないことも多い. そのため, 論文の本文中からセクション名を付随しながら重要な文を抜粋することで, 疑似的に構造化されたアブストラクトが生成できる. また, アブストラクトよりも大きいサイズで要約を出力することでより内容の充実したアブストラクトが生成できるということも考えられる. さらに, 近年の機械学習ベースの抽出型要約の研究ではニュース記事などの比較的小さなサイズのデータセットがよく用いられているが [5, 9, 10], 長いサイズの文書要約研究の取り組みとして良い実験材料にもなる. これはカテゴリーとしては単一文書の抽出型要約に属する.

本研究ではこれらのうち後者の単一論文の抽出型要約を対象タスクとする. このタスクの問題点は, 論文のようにサイズの大きな文書は抽出型要約を適用するにしても単純に抽出候補となる文の数が多いため要約の難易度が高いこと, またニューラルなモデルでもこのような長大な系列を扱うのは難しいことである. 本研究ではこれらの問題へ特にフォーカスを当ててニューラルな抽出型要約モデルを構築することを目的とする.

## 1.2 本研究の提案

本研究では, 学術論文のような長大なサイズの文書を抽出的に要約するためのニューラルネットワークのモデルを提案する.

多くの言語処理タスクではリカレントニューラルネットワーク (Recurrent Neural Network, RNN) という系列情報を有効に処理できるニューラルネットワークがよく用いられている. テキストは一般に文や単語の系列で構成されていると考えられるため, RNN での処理が有効であると考えられる. 論文や Wikipedia の記事のように比較的長いサイズのテキストでは文の系列が必然的に長くなってしまうが, 一方で RNN は入力の系列が長くなると処理の精度が落ちることが知られている [17, 18]. この問題に対し, 文書構造に沿って RNN を階層的に構築することが一つの対策として考えられている [2, 19]. 論文の場合は一般に単語, 文, パラグラフ, セクションという章構造で構成されていることが多い. 本研究では, これらの既存研究に基づき学術論文の章構造に即した階層的 RNN モデルを提案する.

次に, 抜粋候補となる文の数が多いため抽出型要約の難易度が高いという問題に対しても, 章構造を利用する手法を提案する. 抽出型要約を教師有り学習に基づいて行う場合は, あらかじめ本文中の

全ての文に対して正例・負例のラベルを貼ってそれを学習データとしている. 本研究ではそれに加えてパラグラフとセクションにもラベルを設け, モデルが正例文を見つけやすくなるようサポートする. 論文はセクション, パラグラフ, 文をノードとする木構造と捉えることができるが, 正例文を含むパラグラフ, セクションを正例と定義する. あるノードでのラベル予測に親ノードでの予測を組み込むことで, 親ノードが正例である木の中にある文を選ぶようにモデルを誘導する. この手法によって, 同時に負例が親ノードである木に含まれる文, つまり本筋とは関連性の薄い文を候補から外すことも期待できる.

## 1.3 本研究の貢献

本研究の貢献は以下のとおりである.

**学術論文のための階層的ニューラルモデルの提案**

学術論文の文書構造を生かし, 階層的に RNN を適用した抽出型要約モデルを新たに提案した. 提案手法は特徴量設計の負担を減らし, データドリブンなモデルになっている. このネットワーク構造は以下に述べる別の貢献をサポートするのに必要な設計にもなっている.

**セクション・パラグラフ・文の階層的スコア計算手法の提案**

抽出型要約の既存研究は文のみに着目していたが, セクション・パラグラフというより大きな意味のまとまりについてもラベルを付与し, それらのスコアも正しく予測できるよう同時学習するモデルを提案した. また, これらのラベル予測を木構造に沿って活用する分類器も提案した. これにより正例文をより正しく選ぶようになり, 精度が改善されたことを確認した.

**文脈の重要度を意識した文抽出**

直感的には長い文書においては重要な部分と重要でない部分が存在すると考えられるが, 重要性を「アブストラクトとの単語被覆度」という観点で捉えた上で, 正例文が含まれる段落が重要性が高く, 正例文が含まれない段落が重要性が低いことを予備実験で定量的に確認した. そして, 提案モデルが抽出した負例の文はより重要度の高い箇所から選ばれる傾向が高いことを示した.

**適用範囲の広いモデル設計**

本研究で提案した手法はいずれも論文に特有な特徴量を用いていない. 本研究が着目したのは文書構造であるが, これは論文に特有のものではなくある程度長い文書なら一般的に存在する特徴であるため, 他の文書にも適用可能であると考えられる.

## 1.4 本研究の構成

本稿の構成としては以下のとおりである.

**第 2 章　理論**

本研究の提案モデルの基本となっている理論について述べる．具体的には, 教師有り学習の枠組み, ニューラルネットワークの基本的な知識, そして自然言語処理においてよく用いられるニューラルネットワークについて説明する.

**第 3 章　関連研究**

機械学習ベースの抽出型要約および文書処理のための階層的ニューラルモデルの関連研究について述べる.

**第 4 章　提案手法**

本研究の提案モデルの具体的な中身について説明する.

**第 5 章　実験**

提案モデルの有効性を示す実験について述べる．また, データセットの作成, 実験の結果と考察についても報告する.

**第 6 章　結論**

最後に本論文のまとめと今後の課題について述べる.

# 第2章　理論

機械学習には大まかには, 教師有り学習, 教師なし学習, その中間のような考え方の強化学習という枠組みがあるが, 本研究で適用する教師有り学習についてまず説明する. 次にそれを抽出型要約に適用する際にどのように定式化するかを説明する. また, 機械学習のモデルはサポートベクターマシーンや平均化パーセプトロンなど様々なものが存在するが, 以下ではニューラルネットワークについて考えることとし, ニューラルネットワークの基本的な枠組みを述べる. 最後に自然言語処理の分野でよく用いられるリカレントニューラルネットワークと単語のベクトル表現について説明する.

## 2.1　教師有り学習の枠組み

入力とそれに対応する理想的な出力が $N$ 組,

$$\{(\boldsymbol{x}_1, \boldsymbol{y}_1), (\boldsymbol{x}_2, \boldsymbol{y}_2), ..., (\boldsymbol{x}_N, \boldsymbol{y}_N)\}$$

与えられているとして, これらに対して理想的な出力が実現されるようにモデルのパラメーター $\theta$ を調整することを考える. つまりどの $i = 1, 2, ..., N$ に対しても, $\boldsymbol{x_i}$ をモデルに入力したときに $\boldsymbol{y}_i$ が出力されるようにパラメーター $\theta$ を調節する. これを教師有り学習と呼び, 入出力の組をまとめて学習データと呼ぶ. あるタスクについてまとまった量の学習データを用意して, モデルにそのタスクの入出力の法則のようなものを覚えさせ, 学習データに含まれないような未知の入力に対しても望ましい出力を返させるようにするのが目的である. 基本的には学習データが多ければ多いほどそのタスクの普遍的な法則をカバーしやすくなり, 未知のデータに対する精度 (汎化性) も向上することが期待される.

このとき, モデルに $\boldsymbol{x_i}$ を入力したときの出力と, このとき望まれる出力 $\boldsymbol{y}_i$ との違いを表す尺度が大切になる. 出力の間違いが大きければ大きいほどモデルを大きく改良する必要があるし, 逆に小さければあまり更新しない方が良いからである. この尺度を測る関数のことを誤差関数と呼び, 問題設定に応じて適切な関数が用いられる. 例えば回帰タスクにおいては二乗誤差が用いられたり, クラス分類タスクにおいては交差エントロピーが用いられる. 学習においては, この誤差を少なくするようにパラメーターをアップデートする. 具体的には誤差関数を各パラメーターで偏微分した値を計算し, 適切に重みづけしてパラメーターを変更する. このとき, パラメーターの更新には確率的勾配降下法がよく用いられる.

$$\theta \leftarrow \theta - \alpha \frac{\partial E}{\partial \theta} \tag{2.1}$$

ここで，$\alpha$ は学習率と呼ばれパラメーターの更新の大きさを決める．一般に学習率の大きさは人間が決め, その調整は非常に手間がかかるが, 近年では学習率をパラメーターごとに自動で調整するアルゴリズムが提案されている．例えば Adam [20] がその一つである．Adam はよく更新されるパラメーターについては更新の量を小さくし, あまり更新されていないパラメーターについては大きく更新させるという思想に基づいている．このアルゴリズムでは勾配の指数移動平均 $m_t$ と中心化されていない分散の指数移動平均 $v_t$ を用意する．そして，この 2 つは過去の更新の履歴と現時点での更新量から次の量が決定される．

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\frac{\partial E}{\partial \theta} \tag{2.2}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)\frac{\partial E}{\partial \theta}^2 \tag{2.3}$$

ここで $\beta_1$, $\beta_2$ はハイパーパラメーターである．平均を更新に使うことで, 確率的勾配法で与えられる勾配のばらつきを軽減しつつ, 指数移動を使って直近の勾配に相対的に大きな重みを付与することで, 時々刻々のモデルの変化に柔軟な更新をすることができる．ただし，上式は学習の初期は $m_t$ と $v_t$ が初期値 $m_0$ と $v_0$ の影響を強く受けてしまうという問題があるため, 初期値の影響を補正した次の値をパラメータの更新に使っている．

$$\hat{m}_t = \frac{m_t}{1-{\beta_1}^t} \tag{2.4}$$

$$\hat{v}_t = \frac{v_t}{1-{\beta_2}^t} \tag{2.5}$$

以上をまとめて, 式 2.1 の代わりに

$$\theta \leftarrow \theta - \alpha\frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon} \tag{2.6}$$

という更新式を用いる．$\epsilon$ は分母が 0 になってしまうことを防ぐ定数で $10^{-8}$ といった小さい値が採用される．更新式中で勾配の平均を標準偏差 $\sqrt{\hat{v}_t}$ で割ることで, 勾配が大きいパラメータは学習率を小さくし, 勾配が小さいパラメータは学習率を大きくしている．

### 2.1.1 教師有り学習に基づく抽出型要約タスクの定式化

教師有り学習に基づいて抽出型要約を行うときの数学的フレームワークについて説明する．このアプローチでは抽出型要約を二値分類問題として解くこととなる．つまり文書中の各文について，要約に含めるか含めないかの二択を予測するタスクである．モデルは文書が入力されると文書中の全文に対してその文が正例である確率, つまりどれぐらい要約に含めるべきかを表す信頼度を計算する．以降この値をスコアと表現する．モデルの設計者は文書中の全ての文に対し, その文が要約としてふさわしいかふさわしくないかの二値のラベルが貼られた学習データを用意する．この正解ラベルは

理想的には人間の手で作成されるのは望ましいが, 実際的には人間が書いた要約と最も単語が被覆するような文集合を本文中から何らかのアルゴリズムで探索し, それを正例とみなすことが多い. 学習データを通して, システムは計算したスコアが正解ラベルと一致するようにパラメーターを更新していく. 学習を終えたモデルは未知の文書に対して, その文書中で重要な文に高いスコアを付与し, そうでない文に低いスコアを付与することが期待される. 出力の際には単語数が一定の長さに達するまで, 高いスコアの文から順に抜き出していく. また評価については, 各文書についてモデルが抽出した文の集合と, その文書にあらかじめ付随している人間が書いた正解の要約がどれほど類似しているかによって判断する. これらをそれぞれシステム要約と参照要約と呼ぶ. この類似度を測る指標としては, 単語がどれほど一致しているかを見る ROUGE スコア [21] がよく用いられている.

これらを数学的に以下のように定式化する. まず, $x$ をシステムへの入力文書とし, $x$ は $T$ 個の文 $s$ で構成されているものとする. これを, $x = [s_1, s_2, ..., s_T]$ と書くことにする. 次に $y_t \in \{0, 1\}$ を $t$ 番目の文の正解ラベルとし, $y_t = 0$ の場合は出力要約に含めない, $y_t = 1$ の場合は出力要約に含めることを示すものとする. 以降これら二つをそれぞれ負例, 正例と表現することにする. 文書中の全文の正解ラベルは $y = [y_1, y_2, ..., y_T]$ と表される. そして, システムが予測した $s_t$ が正例である確率 $p(y_t = 1|x)$（すなわちスコア）を $p_t$ と記述する. すなわち, $p_t = 0$ ならばシステムはその文が要約として含まれないと判断し, $p_t = 1$ ならば要約として含めると判断することになる. ここで, 文 $s_t$ のスコアが正解ラベル $y_t$ である確率分布は, これが二値分類であることを考慮すると,

$$p(y_t|x) = p(y_t = 1|x)^{y_t} p(y_t = 0|x)^{1-y_t} = p_t^{y_t}(1 - p_t)^{1-y_t} \tag{2.7}$$

と表現できる. モデルのパラメーター $\theta$ は, 学習データを用いてモデルが与える事後分布が, データが与える分布と最も整合するように最尤推定を行う. 一つの学習サンプルについてのモデルのパラメーター $\theta$ に対する尤度は

$$L = \prod_{t\,=\,1}^{T} p(y_t|x) = \prod_{t\,=\,1}^{T} p_t^{y_t}(1 - p_t)^{1-y_t} \tag{2.8}$$

となる. これを最大化するようにしたいので, 逆に負の対数尤度

$$E = -\sum_{t\,=\,1}^{T} \{y_t \log p_t + (1 - y_t) \log (1 - p_t)\} \tag{2.9}$$

を最小化すればよく, これを誤差関数とする.

$\mathcal{D} = \{x_n, y_n\}_{n=1}^{N}$ を学習データとして, $n$ 番目の学習データに対して式 2.9 に従って得られる誤差を $E_n$ とすると, 学習全体としては

$$\frac{1}{N}\sum_{n=1}^{N} E_n \tag{2.10}$$

を最小化するようにモデルのパラメーターを更新するのが目的である.

上の説明では事前に二値の正解ラベルを用意し, モデルが予測したラベルと正解ラベルのギャップを誤差関数とする考えを説明したが, それ以外にも問題設計は存在する. 例えば, 近年の生成型のタスクでは ROUGE スコア [21] を直接的に誤差関数に組み込むという研究もおこなわれている [22, 23, 24].

抽出型要約に対しても式 2.9 の代わりに ROUGE スコアを誤差関数に組み込むことは可能であると考えられる. ただし ROUGE スコアは離散的な指標であり一般に微分不可能であるので, 強化学習を用いて学習する必要がある. また, 要約ではないが単語生成のタスクにおいて, 敵対的生成ネットワーク (Generative Adversarial Network, GAN) を用いた研究もある [13]. GAN はより「人間らしい」出力を目指す枠組みで, 入力が「人間らしい」か否かを二値分類する識別器と, 識別器に「人間らしい」と思わせる生成器の 2 つを交互に学習させる. 生成器は識別器をどれほど「人間らしい」と騙せたかを報酬として強化学習でアップデートされる.

ここで, 図 1.1 で挙げた入力文書を例に抽出型要約を機械学習ベースで処理する場合を説明する. 入力文書 $x = [s_1,\ s_2,\ s_3,\ s_4]^T$ は以下のようになる.

$$\begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix} = \begin{bmatrix} \text{I walked out of the house and got in the car .} \\ \text{It started rough , and I let it idle for a few minutes .} \\ \text{I was late enough so that most of the traffic had cleared off of the freeway .} \\ \text{I pulled into the parking garage under my building at 8 : 45 .} \end{bmatrix}$$

それに対して, 正解ラベル $y = [y_1,\ y_2,\ y_3,\ y_4]^T$ は, 一番目と四番目の文が要約として選ばれるようにしたいので

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

となる.

## 2.2 ニューラルネットワークの基本的な考え

### 2.2.1 パーセプトロンと誤差逆伝搬

ニューラルネットワークは現在様々な種類が存在しているが, ここでは最もシンプルな一層のパーセプトロンを例として説明する. (普通パーセプトロンという名称は多層のパーセプトロンという文脈で使われることが多く，一層のパーセプトロンという表現はあまり一般的ではないが，最も簡単な例して用いるためここではその表現を用いる.) パーセプトロンは以下のように, 一つのベクトルを入力として受け取り, それに行列演算と非線形操作を加えることで出力を吐く.

$$\boldsymbol{o} = f(W\boldsymbol{x} + \boldsymbol{b}) \tag{2.11}$$

ただし, 入力が $\boldsymbol{x}$, それに対するモデルの出力が $\boldsymbol{o}$ である. $\boldsymbol{x}$ をモデルに入力することそのものは特別なことをしてはいないが, この処理を行う部分のことを便宜的に入力層とよぶ. また, モデルの最終的な出力を吐く部分を出力層と呼び, この例ではこの一つのパーセプトロンが出力層となる. つまりこのモデルは全体で見ると二層のニューラルネットワークということになる.(入力層を除いて一層と解釈する考えもあり, 層について明確な定義があるわけではない.) また, $W$ は重み行列, $\boldsymbol{b}$ はバイアス

項と呼ばれいずれもモデルのパラメーターである. また, $f$ は活性化関数と呼ばれ, 一般的には tanh やシグモイド関数などの非線形関数が用いられる. ここで, $\boldsymbol{x}$ に対する理想的な出力を $\boldsymbol{y}$ とし, 誤差関数を二乗誤差

$$E = \frac{1}{2}||\boldsymbol{o} - \boldsymbol{y}||^2 \tag{2.12}$$

とすると, 学習データ中のサンプルを受け取り, その都度 $\dfrac{\partial E}{\partial W}$ および $\dfrac{\partial E}{\partial \boldsymbol{b}}$ を求め, 式 2.1 に従い $W$ と $\boldsymbol{b}$ をそれぞれ更新することになる. ここで, 式 2.11 を

$$\boldsymbol{a} = W\boldsymbol{x} + \boldsymbol{b} \tag{2.13}$$

$$\boldsymbol{o} = f(\boldsymbol{a}) \tag{2.14}$$

と分解すると

$$\frac{\partial E}{\partial \boldsymbol{o}} = \boldsymbol{o} - \boldsymbol{y} \tag{2.15}$$

$$\frac{\partial E}{\partial \boldsymbol{a}} = \frac{\partial E}{\partial \boldsymbol{o}}\frac{\partial \boldsymbol{o}}{\partial \boldsymbol{a}} = (\boldsymbol{o} - \boldsymbol{y}) \odot f'(\boldsymbol{a}) \tag{2.16}$$

のように, 出力層に近い順に偏微分が求まる. さらに, チェインルールにより

$$\frac{\partial E}{\partial W} = \frac{\partial E}{\partial \boldsymbol{a}}\frac{\partial \boldsymbol{a}}{\partial W} = \{(\boldsymbol{o} - \boldsymbol{y}) \odot f'(\boldsymbol{a})\}\boldsymbol{x}^T \tag{2.17}$$

$$\frac{\partial E}{\partial \boldsymbol{b}} = \frac{\partial E}{\partial \boldsymbol{a}}\frac{\partial \boldsymbol{a}}{\partial \boldsymbol{b}} = (\boldsymbol{o} - \boldsymbol{y}) \odot f'(\boldsymbol{a}) \tag{2.18}$$

とパラメーターの偏微分が求まる.

ここで, パーセプトロンが二層の場合も同様のルールにより偏微分が伝搬する. それぞれのパーセプトロンの計算は以下のようであるとする.

$$\boldsymbol{h} = g(W_1\boldsymbol{x} + \boldsymbol{b}_1) = g(\boldsymbol{a}_1) \tag{2.19}$$

$$\boldsymbol{o} = f(W_2\boldsymbol{h} + \boldsymbol{b}_2) = f(\boldsymbol{a}_2) \tag{2.20}$$

一つ目のパーセプトロンはモデル内に組み込まれた層であり, 外部からは観測できないため隠れ層と呼ばれる. また, 隠れ層の出力 $\boldsymbol{h}$ を隠れ状態ベクトルと呼ぶ. この場合は入力層, 隠れ層, 出力層の全てをまとめて見ると三層のニューラルネットワークということになる. 出力層における逆伝搬は式 2.17-2.18 と同様に計算できる. 隠れ層と出力層は隠れ状態ベクトル $\boldsymbol{h}$ を介して接続されているが, $\boldsymbol{h}$ の偏微分は

$$\frac{\partial E}{\partial \boldsymbol{h}} = \frac{\partial E}{\partial \boldsymbol{a}_2}\frac{\partial \boldsymbol{a}_2}{\partial \boldsymbol{h}} = W_2^T\{(\boldsymbol{o} - \boldsymbol{y}) \odot f'(\boldsymbol{a}_2)\} \tag{2.21}$$

のように計算できる. これが求まれば隠れ層のパーセプトロンのパラメーターも式 2.17-2.18 と同様に計算できる. このようにあるサブネットワークに関して, 出力ベクトルの偏微分から入力ベクトルの偏微分が計算されれば, 前段のサブネットワークに偏微分が伝搬される. つまり, 最終層から順番に偏微分を計算していくことで効率よく全ての偏微分が求まる. これを誤差逆伝搬法という.

## 2.3 自然言語処理におけるニューラルネットワーク

### 2.3.1 単語の分散表現

自然言語処理のタスクでは一般に単語ないし文字が処理する対象の最小単位となる. ここでは単語が最小単位であるとすると, 計算機上で単語を文字列型として扱うのは計算コストが増してしまうため, メインの処理を始める前に対象となるデータセット中をくまなく走査し, 全ての単語を数字に置き換えることが多い. 図 1.1 の文書を例にとると“I” $\to 0$, “walked” $\to 1$, “out” $\to 2$, ... のように出現するすべての単語が順に番号に置き換えられる. この対応表を辞書 $\boldsymbol{V}$ と呼ぶ. このとき各単語は, $|\boldsymbol{V}|$ 次元で, その単語を指すインデックスに対応する要素だけ 1 で, それ以外のすべての要素が 0 となる疎なベクトルで表現できる. これを one-hot ベクトルと呼ぶ. one-hot ベクトルを $\boldsymbol{e}$ と書くことにすると, 例えば先ほどの“out”の one-hot ベクトルは $\boldsymbol{e}(\text{“out”}) = [0,\ 0,\ 1,\ 0,\ ...,\ 0,\ 0]^T$ となる. 同様の考えで文中に含まれる全て単語の one-hot ベクトルを足し合わせることでできる文のベクトルを bag-of-words という. one-hot ベクトルも bag-of-words も単にデータセット中の単語を数字に置き換えたものの集合に過ぎないので, 意味情報を全く持っていない.

近年では意味情報を考慮した上で単語を数百次元のベクトル空間に写像する研究が行われており [25], 多くのタスクで取り入れられている. 意味的に近い単語をベクトル空間内で近い位置に分布するように写像することで, 例えば単語ベクトル同士のコサイン距離を測ることで定量的に意味の近さ・遠さを測量できるといった利点がある. このような単語ベクトルの表現を単語の分散表現や word embedding などといい, データセット中の単語ベクトルを辞書の番号順に行方向に並べて出来る行列を embedding matirix などという. word embedding を $\boldsymbol{v}$, embedding matirix を $\boldsymbol{W}_{embed}$ と書くことにすると, 図 1.1 の文書の例においては

$$W_{embed} = \boldsymbol{v}(\text{“I”}) \oplus \boldsymbol{v}(\text{“walked”}) \oplus \boldsymbol{v}(\text{“out”}) \oplus \cdots \oplus \boldsymbol{v}(\text{“8 : 45”}) \tag{2.22}$$

となる. ただし, $\oplus$ はベクトルを行方向に連結する演算を表す. また辞書中の任意の単語 $w$ について one-hot ベクトル $\boldsymbol{e}(w)$ と embedding matrix $W_{embed}$ を用いて対応する単語ベクトル $\boldsymbol{v}(w)$ が取り出せる.

$$\boldsymbol{v}(w) = W_{embed}\boldsymbol{e}(w) \tag{2.23}$$

embedding matrix は自然言語処理の様々なタスクの基礎的なモデルパラメータとなっている. また, ニューラルネットワークと親和性が高いこともメリットで, 後続のニューラルネットワークと計算ノードを接続することで embedding matrix 中の全ての単語ベクトルに対しても誤差逆伝搬で勾配を計算することができる. つまり, 学習を通じてそのタスクに最適な単語ベクトル表現を獲得できる. ただし, データセット中の全ての単語のベクトルをパラメーターとして保持するとモデルパラメー－

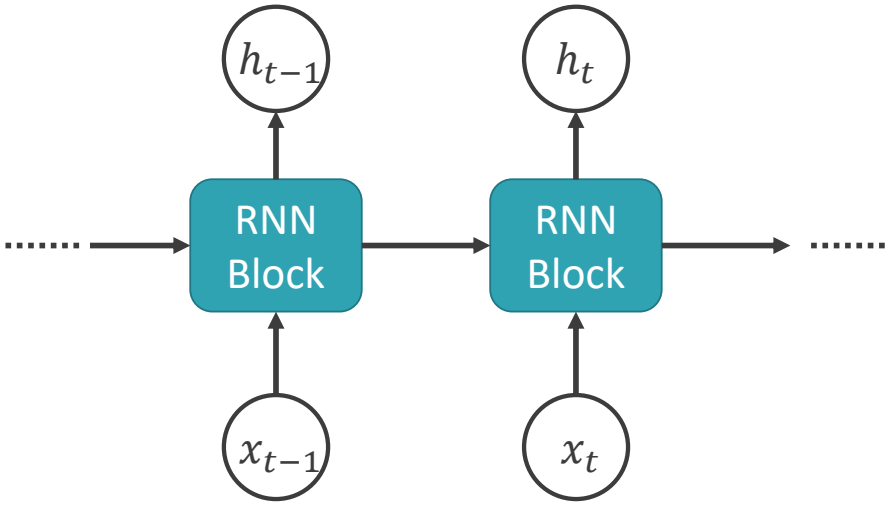


図 2.1: RNN の時系列処理を時間方向に展開した図. $\boldsymbol{x}_t$, $\boldsymbol{h}_t$ はそれぞれ時刻 $t$ における入力ベクトル, 隠れ状態ベクトルである.

のサイズが大きくなりすぎたり, 計算コストが増大化するため, 登場頻度の低い単語はすべて一括して*UNK*(unkown) という一つの単語に置き換えてしまうことも多い.

### 2.3.2 リカレントニューラルネットワーク

2.2.1 節で説明したパーセプトロンは基本的に一つのベクトルを入力として受け取り, 一つのベクトルを出力として吐き出すものであった. それに対し, リカレントニューラルネットワーク (Recurrent Neural Network, RNN) は可変長の入力系列, つまり複数の入力ベクトルの連なりを扱うのに適したモデルである. 系列を前から数えたときのインデックスを時刻と呼ぶことにすると, RNN は時刻 $t$ で, その時刻に対応する入力ベクトル $\boldsymbol{x}_t$ と時刻 $t-1$ における RNN の出力 $\boldsymbol{h}_{t-1}$ を受け取り, それに対応する出力 $\boldsymbol{h}_t$ を吐き出す. RNN はやろうと思えばいつまでも入力を受け取り続けられるので, 特に終端はない. つまり明確に出力層はなく, ある意味全ての時刻の処理が隠れ層であり, あらゆる時刻の出力が隠れ状態ベクトルである. (ただし, 入力系列の長さは事前にわかっていることが多いので, 実質的には系列の終端時刻における隠れ状態ベクトルがその系列全体に対する出力ベクトルとみなせる.) 図 2.1 に RNN の処理の様子を示す. RNN は前の時刻の隠れ状態ベクトルと現時刻の入力ベクトルを使って, 現在の隠れ状態ベクトルを更新するため, 理想的には入力履歴を考慮した出力が得られる. 定性的にみると, 時刻 1 から $t$ までの入力系列 $[\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_t]$ の情報を一つの固定長ベクトルに圧縮したものが $\boldsymbol{h}_t$ である. 言語処理においては単語や文をベクトルで表現して, それらを系列として RNN で処理する. 同一のモデルで任意の長さの系列を扱えるのが RNN の長所であり, 特に文書や文などはサンプルによって中に含まれている文や単語の数は一般に異なるため, RNN は適しているといえる.

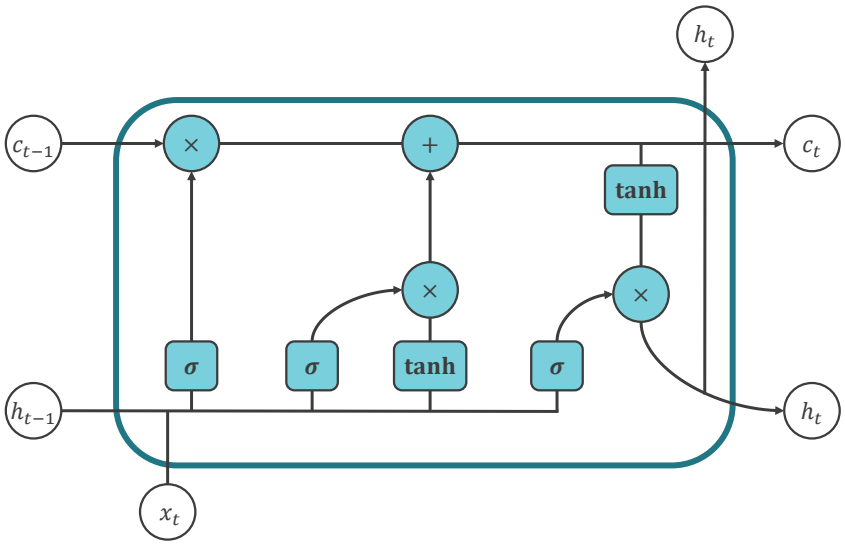


図 2.2: 時刻 $t$ における LSTM の処理.

一方で, RNN は時間方向に深いため, 時間的に離れている時刻に発生した誤差を伝搬させるのは難しい. つまり長期記憶は難しく, 直近の依存関係しか学習できないことが知られている. そのため, ゲートを設けて長期記憶と短期記憶のバランスを習得する手法がいくつか提案されている. 本節ではそのうち超短期記憶 (Long Short-Term Mermory, LSTM) [26] を説明する. 時刻 $t$ における LSTM の処理は以下のとおりである.

$$\boldsymbol{i}_t = \sigma(W^{(i)}\boldsymbol{x}_t + U^{(i)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(i)}) \tag{2.24}$$

$$\boldsymbol{f}_t = \sigma(W^{(f)}\boldsymbol{x}_t + U^{(f)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(f)}) \tag{2.25}$$

$$\boldsymbol{o}_t = \sigma(W^{(o)}\boldsymbol{x}_t + U^{(o)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(o)}) \tag{2.26}$$

$$\boldsymbol{u}_t = \tanh(W^{(u)}\boldsymbol{x}_t + U^{(u)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(u)}) \tag{2.27}$$

$$\boldsymbol{c}_t = \boldsymbol{i}_t \odot \boldsymbol{u}_t + \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} \tag{2.28}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t) \tag{2.29}$$

ここで, $\sigma$ はシグモイド関数, $\odot$ は要素積を表し, $W^{(\cdot)}$ および $U^{(\cdot)}$ は重み行列, $\boldsymbol{b}$ はバイアス項を表す. また, $\boldsymbol{i}_t$ は入力ゲート, $\boldsymbol{f}_t$ は忘却ゲート, $\boldsymbol{o}_t$ は出力ゲート, $\boldsymbol{u}_t$ は更新用のメモリセル, $\boldsymbol{c}_t$ はメモリセルとされ, それぞれが掛け合わさることによって系列情報をユニットがどの程度記憶しておくかということを制御し, 長い時系列情報を処理している. ただし, $\boldsymbol{h}_0$ はゼロベクトルとする. 以降, 簡単のため式 2.24 から式 2.29 までの一連の処理を $\boldsymbol{h}_t = \mathrm{LSTM}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1})$ と書くことにする. これらの操作の様子を図 2.2 に示す.

# 第3章　関連研究

本章では Cheng と Lapata による抽出型の要約を機械学習ベースで行う研究 [9] を中心に説明する. Cheng と Lapata のモデルは機械翻訳で最初に提案されたエンコーダー・デコーダーモデルを下敷きにしているため, まずエンコーダー・デコーダーモデルによる機械翻訳の基本的な仕組みを説明したのち, Cheng と Lapata のモデルについて説明する. 次に, 文書を対象としたニューラルモデルの関連研究について述べる.

## 3.1　機械翻訳

機械翻訳は簡単にいうとある言語の単語の系列をシステムに入力し, それを他言語で言い換えたときの単語の系列を出力するタスクである. 例えば英日翻訳であれば,

$$[\text{this}\quad \text{is}\quad \text{a}\quad \text{pen}\quad .\,] \longrightarrow [\text{これ}\quad \text{は}\quad \text{ペン}\quad \text{です}\quad \text{。}]$$

という変換を施す. RNN が系列処理に適していることは先に述べたが, 近年の機械翻訳では入力言語側の単語系列を読み込んで固定長のベクトルにマッピングする RNN と, それを受け取って対象言語の単語系列を順に出力する RNN の二つでモデルが構成されていることが多い [27]. 前者の RNN をエンコーダー, 後者の RNN をデコーダーと呼び, これら二つをベースにしたモデルをエンコーダー・デコーダーモデルという. 機械翻訳は自然言語処理の中でも最も研究されている分野のひとつであり, 精度を向上させる様々な機構が提案されているが本章では最もシンプルなモデルについて説明する. また, ここでの RNN は LSTM とする. 先程の例でいうと, モデル全体への入出力の対 $(\boldsymbol{x}, \boldsymbol{y})$ は

$$\begin{aligned}\boldsymbol{x} &= [\boldsymbol{e}(\text{“this”}), \boldsymbol{e}(\text{“is”}), \boldsymbol{e}(\text{“a”}), \boldsymbol{e}(\text{“pen”}), \boldsymbol{e}(\text{“.”}), \boldsymbol{e}(\text{“EOS”})] \\ \boldsymbol{y} &= [\boldsymbol{e}(\text{“ これ ”}), \boldsymbol{e}(\text{“ は ”}), \boldsymbol{e}(\text{“ ペン ”}), \boldsymbol{e}(\text{“ です ”}), \boldsymbol{e}(\text{“ 。 ”}), \boldsymbol{e}(\text{“EOS”})]\end{aligned}$$

となる. どこまでが入力系列・出力系列の終わりかをモデルが理解しやすくするため, 両系列の終わりに終端を知らせる特殊な単語 (End of Sequence, EOS) を付与することが多い.

より一般的に入力単語系列を $x = [x_1, x_2, ..., x_{T_{src}}]$, 出力単語系列を $[y_1, y_2, ..., y_{T_{tgt}}]$ とすると, モデルはある入力系列に対して, 対応する出力単語系列が成立する確率 $p(y_1, y_2, ..., y_{T_{tgt}}|x)$ が最大化す

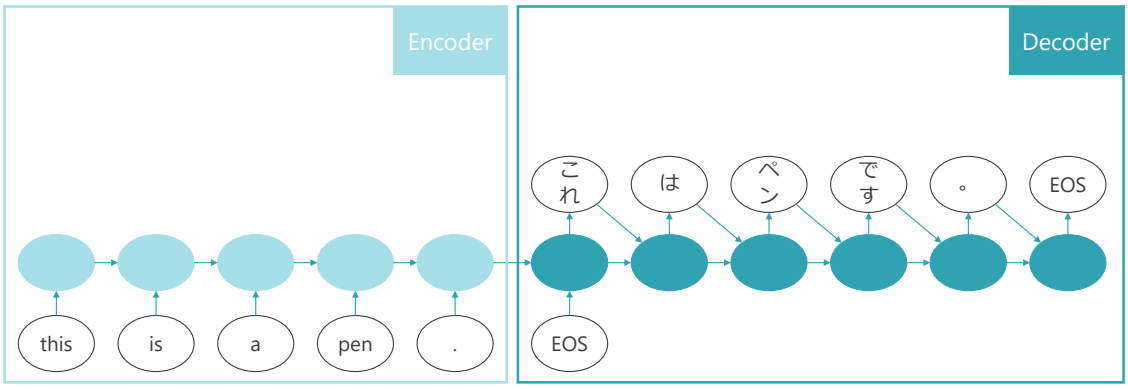


図 3.1: 機械翻訳の例.

るように学習する. そのため, 誤差関数としてはこの確率の負の対数尤度をとったものとなる.

$$\begin{aligned} E &= -\log p(y_1, y_2, ..., y_{T_{tgt}}|x) \\ &= -\log \prod_{t=1}^{T_{tgt}} p(y_t|y_1, ..., y_{t-1}, x) \\ &= -\sum_{t=1}^{T_{tgt}} \log p(y_t|y_1, ..., y_{t-1}, x) \end{aligned} \tag{3.1}$$

エンコーダー側の処理は入力単語のベクトル系列を読み込んで, 隠れ状態ベクトルに圧縮することである.

$$\boldsymbol{h}_t = \mathrm{LSTM}_{enc}(\boldsymbol{v}(x_t), \boldsymbol{h}_{t-1}) \tag{3.2}$$

デコーダー側の処理は系列情報を組み込みながら各時刻で単語の確率分布を計算することである. 式 3.1 に従うと学習の段階では, 時刻 $t$ においては $y_1, ..., y_{t-1}$ が与えられているものとして計算する. よってデコーダー側の時刻 $t$ に該当する LSTM には正解単語である $y_{t-1}$ のベクトルを入力する.

$$\bar{\boldsymbol{h}}_t = \mathrm{LSTM}_{dec}(\boldsymbol{v}(y_{t-1}), \bar{\boldsymbol{h}}_{t-1}) \tag{3.3}$$

$$\boldsymbol{o}_t = \mathrm{softmax}(W\bar{\boldsymbol{h}}_t + \boldsymbol{b}) \tag{3.4}$$

ここで softmax はソフトマックス関数を表し, $\boldsymbol{o}_t \in \mathbb{R}^{|V_{tgt}|}$ は単語の確率分布を表す. ただし $|V_{tgt}|$ はターゲット言語の辞書の大きさを表す.

ここで, 時刻 $t$ においてある単語 $w$ が出力される確率は

$$p(w|\boldsymbol{x}) = \boldsymbol{e}(w)^T \boldsymbol{o}_t \tag{3.5}$$

となるが, テストの段階では正解の単語は当然分からないので 1 時刻前に予測した単語のベクトルをデコーダーの LSTM に入力する.

$$w_{t-1}^{g} = \mathrm{argmax}_{w} p(w|\boldsymbol{x}) \tag{3.6}$$

$$\bar{\boldsymbol{h}}_{t} = \mathrm{LSTM}_{dec}(\boldsymbol{v}(w_{t-1}^{g}), \bar{\boldsymbol{h}}_{t-1}) \tag{3.7}$$

この様子を図 3.1 に示す.

## 3.2 教師有り学習に基づく抽出型要約モデル

従来は抽出型要約に機械学習を適用する場合は人手で特徴量を考案する手法が主であった. 例えば文の位置や長さ, 文中に含まれる単語などである. しかし, 近年では Cheng と Lapata [9] がよりニューラルネットワークをディープに接続したパワフルなモデルを提案した. 後続の研究もそれに続いて抽出型要約のニューラルモデルを提案している. Nallapati ら [5] は単語系列を扱う層と文系列を扱う層の二層を用意し, かつそれぞれの層では RNN が双方向に走っているようなモデルを提案した. そのようにして文及び文書全体のベクトル表現を獲得し, それらにプラスして文の位置や既に含んだ文の内容なども特徴量に使って二値分類を行うモデルになっている. Isonuma ら [10] は Cheng と Lapata のモデルをベースに, 文書分類を行うネットワークを接続し二つのタスクを同時に学習することで, 参照要約が付随していないようなデータセットに対しても要約が実現するようなモデルを提案した. Narayan ら [11] も Cheng と Lapata のモデルをベースに, 画像付きニュース記事の画像キャプション情報を特徴量として組み込んだ要約モデルを提案した. 以下, Cheng と Lapta のモデルについて説明する.

### 3.2.1 Neural Network Sentence Extracor

Cheng と Lapata [9] は抽出型要約のためのエンコーダー・デコーダーモデルを構築した. これを Neural Network Sentence Extractor(NN-SE) と呼ぶ. 従来的には, 機械学習に基づく抽出型要約は特徴量工学に基づく手法が多かったが, 本研究はよりデータドリブンなモデルになっているのが特徴である. このモデルでは, 入力を文の系列, 出力を各文のスコアの系列と捉え, 文のベクトルの系列をまずエンコーダーで読み込んだあと, デコーダー側で各文のスコアを算出する仕組みになっている. すなわち, モデルは文書 $x = [s_1,\ s_2,\ ...,\ s_T]$ を入力として受け取り, スコアの系列 $[p_1,\ p_2,\ ...,\ p_T]$ を出力する. この処理過程を図 3.2 に示す.

まず文書中に含まれる各文について畳み込みニューラルネットワークを用いてベクトル化したのち, エンコーダー側で LSTM を使って文書中の全文のベクトルを順に入力し, 文書ベクトルを構成する. デコーダー側ではエンコーダー側の情報を基に文書中の全文のスコア $p_t$ $(0 \leq p_t \leq 1)$ を順に計算していく. デコーダー側の LSTM は過去に予測されたスコアが入力されていくため, これまでにどの文が高いスコアを出したか, つまり要約に含まれているかを考慮しながら次の一文のスコアを予測するようになっている. 古典的な抽出型要約の研究においても, 各文のスコアをつける際にこれまで抽

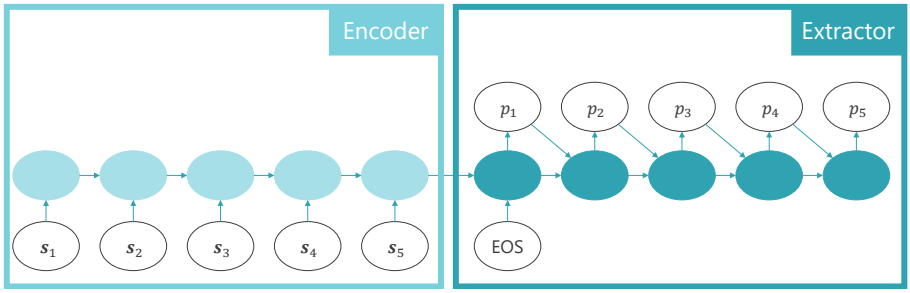


図 3.2: Cheng と Lapata のニューラル要約モデル.

出してきた文を表層的なレベルで考慮する機構は取り入れられていたが, 本研究ではそれらをニューラルネットワークに管理させるといえる. 文書中の文のベクトルを $[\boldsymbol{s}_1, \boldsymbol{s}_2, ..., \boldsymbol{s}_T]$, エンコーダー側の隠れ状態ベクトルを $[\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_T]$, デコーダー側の隠れ状態ベクトルを $[\bar{\boldsymbol{h}}_1, \bar{\boldsymbol{h}}_2, ..., \bar{\boldsymbol{h}}_T]$ と書くとすると, デコーダー側の計算は

$$\bar{\boldsymbol{h}}_t = \mathrm{LSTM}_{dec}(p_{t-1}\boldsymbol{s}_{t-1}, \bar{\boldsymbol{h}}_{t-1}) \tag{3.8}$$

となる. エンコーダー・デコーダーモデルではアテンションと呼ばれる仕組み [17, 28] を用いることが多い. アテンションとは簡単にいうとデコーダー側の処理過程でエンコーダー側の隠れ状態ベクトルを参照する手法である. 機械翻訳においてはエンコーダー側とデコーダー側で扱う言語が異なり, 語順が変わるため, デコーダー側である単語を予測するときに, エンコーダー側のどの部分を翻訳しているのかを推定することが大きな手助けになる. 本モデルにおいてもアテンション機構が組み込まれているが, 図 3.2 のようにエンコーダー側の隠れ状態ベクトルとデコーダー側の隠れ状態ベクトルは一対一に対応しているため, スコア $p_t$ を計算するときはエンコーダー側の $t$ 番目の隠れ状態ベクトルが直接参照される.

$$p_t = \mathrm{MLP}([\boldsymbol{h}_t; \bar{\boldsymbol{h}}_t]) \tag{3.9}$$

ここで, ; はベクトルを列方向に結合する操作を表し, MLP は多層パーセプトロン (Multi Layer Perceptron, MLP) を表す.

本モデルは機械翻訳のようにシーケンス出力として正解ラベルを予測するため, 誤差関数は式 3.1 と同様に考えて次のようになる.

$$E = -\sum_{t=1}^{T} \log p(y_t | y_1, ..., y_{t-1}, x) \tag{3.10}$$

この式に従うと学習時に $t$ 番目のスコアを計算するときは $y_1, ..., y_{t-1}$ が与えられているものとして計算する. すなわち, 式 2.7 における $p(y_t|x)$ を $p(y_t|y_1, ..., y_{t-1}, x)$ に置き換えて考える. よって学習

時のデコーダー側の時刻 $t$ に該当する LSTM の計算は

$$\bar{\boldsymbol{h}}_t = \mathrm{LSTM}_{dec}(y_{t-1}\boldsymbol{s}_{t-1}, \bar{\boldsymbol{h}}_{t-1}) \tag{3.11}$$

$$= \begin{cases} \mathrm{LSTM}_{dec}(\mathbf{0}, \bar{\boldsymbol{h}}_{t-1}) & \text{if} \quad y_{t-1} = 0 \\ \mathrm{LSTM}_{dec}(\boldsymbol{s}_{t-1}, \bar{\boldsymbol{h}}_{t-1}) & \text{if} \quad y_{t-1} = 1 \end{cases} \tag{3.12}$$

となる.

## 3.3 階層的なニューラルモデルに関する研究

近年のニューラルモデルでは文書のような巨大で複雑な情報をいかにして効率よく扱うかという研究が盛んに行われている. 単語や文レベルでは良いベクトル表現が獲得できるようになってきたものの, パラグラフレベルに情報が詰まってくると, いかに深層学習モデルが優れているとはいえ良いベクトル表現を獲得することは簡単ではないと考えられている. そのための一つの対策として, 文書の構造に沿った階層的な構造を取り込んだモデルが提案されている.

Li ら [2] は文書を読み込ませてそのまま入力と同じ内容をモデルに出力させるというタスクにおいて, 文書の構造に則ってエンコーダーおよびデコーダーを階層的に構成したオートエンコーダーを提案し, 精度を向上させたという研究を報告している. このモデルを図 3.3 に示す. このモデルでは, “Mary was hungry . she didn't find any food” という文書が入力されたときに, すべての単語を一つの RNN で処理するのではなく, まず “ Mary was hungry . ” という文を一層目の RNN で処理したのち, 一度その RNN の内部状態をリセットし, 二文目の “ she didn't find any food” を読み込む. 意味合いとしては一層目の RNN は単語のベクトルの系列をエンコードする役割を担っているといえる. そののち, 二つの文それぞれにおいて, 一層目の RNN の文末にあたるユニットの隠れ状態ベクトルを二層目の RNN に渡して処理させる. 二層目の RNN は文のベクトルの系列をエンコードする役割を担っているといえる. そのようにして, 文書全体の内容を読み込んだのち, デコーダーでもまず一層目の RNN で文のベクトルを再構成し, 二層目の RNN で単語を出力している. このように意味のまとまりに即した RNN の構造が文書の処理に効果的に働くことがわかった.

また, 同様の研究として Yang ら [19] は文書分類のための階層構造モデルを構築した. この研究では文書を内容に応じてクラス分けするタスクを行っている. このモデルでも同様に単語のレイヤーと文のレイヤーを担当する RNN に分けられ, ボトムアップにベクトルが獲得される. 特筆すべき点として, 各層の RNN は順方向と逆方向に系列が入力される Bidirectional RNN を用い, さらに単に系列の末尾や先頭の隠れ状態ベクトルを上に流すのではなく, 系列の構成要素全ての隠れ状態ベクトルを適切に重みづけしてベクトルを構成し上の層に渡す仕組みになっている. この重みづけの係数はいわゆる隠れ変数として計算され, 明示的に教師信号は与えられない. 係数を決定するネットワークもモデルの一部として同時学習され, 最適な重みづけが計算できるようにチューニングされる.

また, 直接的には文書を対象にとしていないが, 階層構造を意識した関連研究として Tai ら [29] の木構造に特化した LSTM モデルがある. 通常の直列つなぎの LSTM に比べて, 構文木や係り受け木などの文の階層構造を直接組み込めることが利点である. 文間の関係性を測るタスクおよび単一文の評判分析のタスクにおいて高い精度を達成した.

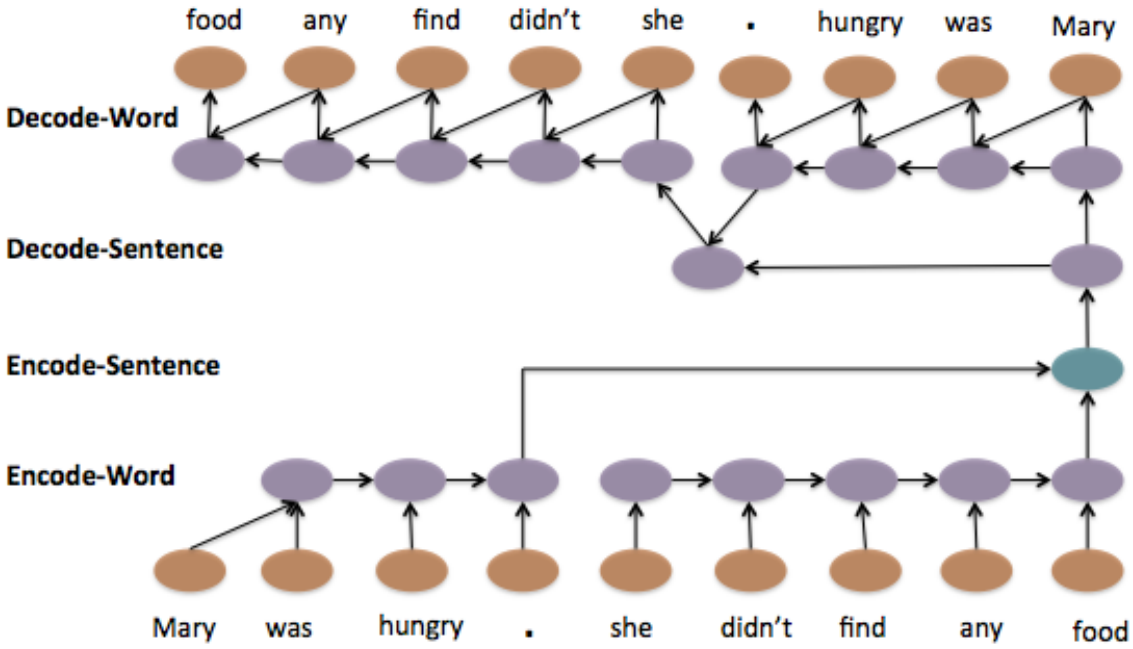


図 3.3: Li らの階層的ニューラルモデル [2].

その他の文書を対象とした研究としては, 文書をいくつかのテキストに分割して, それぞれのテキストで並列に RNN を走らせて最後に情報をまとめるといった手法や [30], RNN ではなく畳み込みニューラルネットワークを用いて長い文書からより良い隠れ状態ベクトルを獲得するという手法 [18] が提案されている.

# 第4章　提案モデル

本章では提案モデルについて述べる．まず, 論文の要約に関する問題点とそれに対する提案手法の概要を述べ, 次に提案モデルの具体的な中身について説明する．

## 4.1　提案手法の概要

### 4.1.1　階層的な RNN モデルの構築

自然言語処理において, 近年の深層学習ベースのモデルでは単語や文のベクトル表現をニューラルネットワークによって獲得し, それらのベクトルを用いて処理を行うのが基本となっている．文抽出型の要約については文を抽出することが一番の目的であるので, 文のベクトルを獲得して後続のニューラルネットワークで文のスコアを計算するのが基本的な流れとなる [5, 9, 10]．文や段落, 文書は構造として系列データとして構成されているため, 要約に限らず機械翻訳や質問応答など多くのタスクでも, 一般に RNN が用いられることが多い [27]．(RNN は系列を逐次的に処理するために計算が並列化できない等の理由で, 近年では畳み込みニューラルネットワークを活用した研究も盛んに行われているものの, 依然基本モデルとして RNN が利用されていることが多い.) 一方で, RNN は入力系列長が長くなればなるほど獲得できるベクトル表現が劣化し, 性能が落ちることが知られている [17, 18]．学術論文をニューラルモデルで抽出的に要約する際の問題点の一つは, 入力文書のサイズが大きいため RNN に読み込ませる系列長が長くなってしまい質の良い文のベクトル表現が獲得しにくいことである．それに対し, 文書を RNN で取り扱う場合は RNN を何層か階層的に構築し, ボトムアップに情報をくみ上げると精度が上がることが知られている [2, 19]．今回のタスクの場合, 論文はセクション, パラグラフ, 文という異なる粒度のまとまりで構成されていると考え, 図 4.1 のような木構造に展開できると仮定し, Li らのモデル [2] と Cheng と Lapta のモデル [9] を参考にこの木構造に沿った階層的なエンコーダー・デコーダーモデルを提案する。また, この木構造に沿って, 文の親ノードにあたるパラグラフやセクションをそれぞれ親パラグラフ, 親セクションなどと書くことにする．

### 4.1.2　周辺情報の参照

もう一つの問題は, 一つの論文中に内容が十分豊富に含まれていてかつ話が色々展開するために, 文中のどのあたりが重要なのかモデルが判別しにくいことである．通常, 文書で正例となる文は中心的なことを述べている文であるはずなので, その文書の中心トピックにまつわるキーワードの単語を含んでいることが予想される．ある程度短い文書なら重要単語を含んでいる文をそのまま選べば正

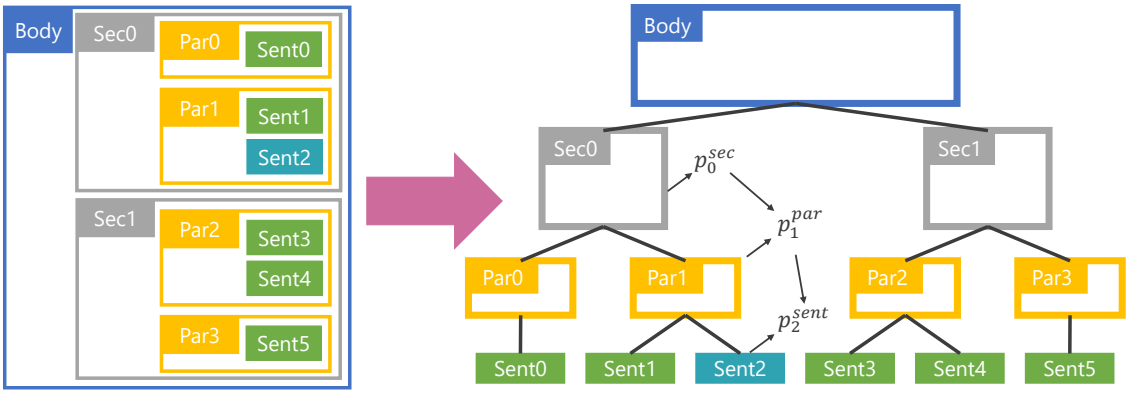


図 4.1: 文書構造に基づく学術論文の木構造化および階層的スコア計算. エメラルドグリーン色のブロックは正例文を表す.

解できる. 実際, 3.2.1 節で述べた通り, Cheng と Lapata のモデルはスコアを決定する際にエンコーダー側から該当する文の隠れ状態ベクトルを直接用いていた. RNN の隠れ状態ベクトルは多少前の情報を含んでいるとはいえ, 基本的には文のみに着目してスコアリングしていることになる. また, 従来研究の多くも基本的には文のみに着目している. しかし論文ほど長い文書になるといたるところでキーワードが使われるため, 多くの負例文もキーワードを含んでいることになり, 文だけ個別に見ていても正例文と負例文の区別がつきづらいと予想される. そのため, 効率的に周辺の文脈の情報を付与し, 参照させる必要があると考えられる. 例えば正例文の周辺には内容的に正例文に関連していて, かつキーワードも含んでいる文があるはずなので, 正例文が含まれる段落には多くのキーワードが登場する. 逆に全く負例文しかないような段落ならば, 個々の負例文にキーワードが散発的に登場しても, その段落全体を見ると重要語はそこまで出てこないはずである. 要するに, 個別に文を見ていても区別はつきにくいが, ある程度文をまとまりとしてみれば重要さの区別がつきやすいのではないかと考えられる. これを踏まえて, 文のラベルを予測する際に, その文が所属するパラグラフおよびセクションの情報も参照するアテンション機構を組み込んだ分類器を構築することを提案する. 前節で階層的にニューラルネットワークモデルを構築することを述べたが, それによってパラグラフおよびセクションのベクトル表現が効率よく獲得できるので, その情報も有効活用できると考えられる.

### 4.1.3 木構造に沿ったスコア計算

教師有り学習では文に対して正解ラベルを設けてそれを抽出させるようにモデルを学習させるが, ディープなニューラルネットワークモデルといえども数百文の中から十数文の正解文をピンポイントで当てるのは難しい. そこで, 4.1.2 節で述べた話に上乗せして, 文だけでなくパラグラフとセクションにも正解ラベルを付与し, これら二つについてもスコアを計算する手法を提案する. パラグラフとセクションの正例・負例の定義は, いずれも子ノードの中に一つでも正例文が含まれていれば正例, 一

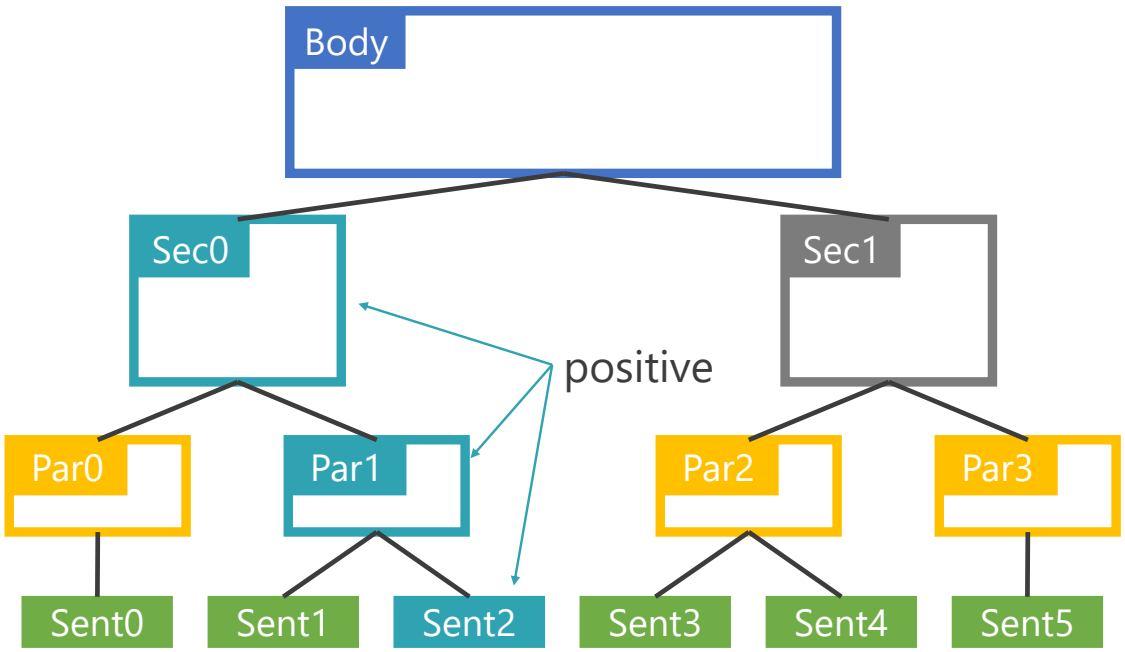


図 4.2: パラグラフおよびセクションの正例・負例の定義の例. この場合, 文 2 が正例であるが, 文 2 を含んでいるパラグラフ 1 およびセクション 0 を正例とラベル付けする. パラグラフ 0, パラグラフ 2, パラグラフ 3 およびセクション 1 は全て負例とする.

つも含まれていなければ負例とする. 図 4.2 に例を示す. ただし, 本研究における正例文・負例文の定義としては, アブストラクトと最もバイグラム（連続する二単語）が一致するような文集合を動的計画法で探索し, その文集合に含まれる文を正例, 含まれない文を負例としている. これについては後の節で詳しく述べる.

従来手法ではいきなり文のノードに対してスコアを計算していたが, そうではなく, 文のスコアを計算するのと同じようにパラグラフおよびセクションについてもスコアを計算し, あるノードのスコアを計算するときにその親ノードのスコアを考慮に入れながら計算することで, よりモデルが正しい木の中の文を選べるように誘導する. 最終的に欲しいのは文のスコアだが, それを計算するときには親セクションおよび親パラグラフのスコアも考慮に入れることで, より正例文を選びやすくするように補正をかける. 定性的に解釈すると, 文章を一回読んだ後, いきなり一個一個の文について予測するのではなく, まずセクションの重要度を考え, 次にその中で各パラグラフの重要度を考え, 最後にその中のセンテンスの重要度を考える, というように徐々にスコープを狭めながらスコアを計算していくことになる. 図 4.1 にそのスコア計算の様子を示す.

また, パラグラフおよびセクションのスコアを計算することの副産物として質の悪い誤りを除去する効果も期待できる. モデルが正例文集合を完璧に当てるのは難しく, いくつかは必ず負例の文, つ

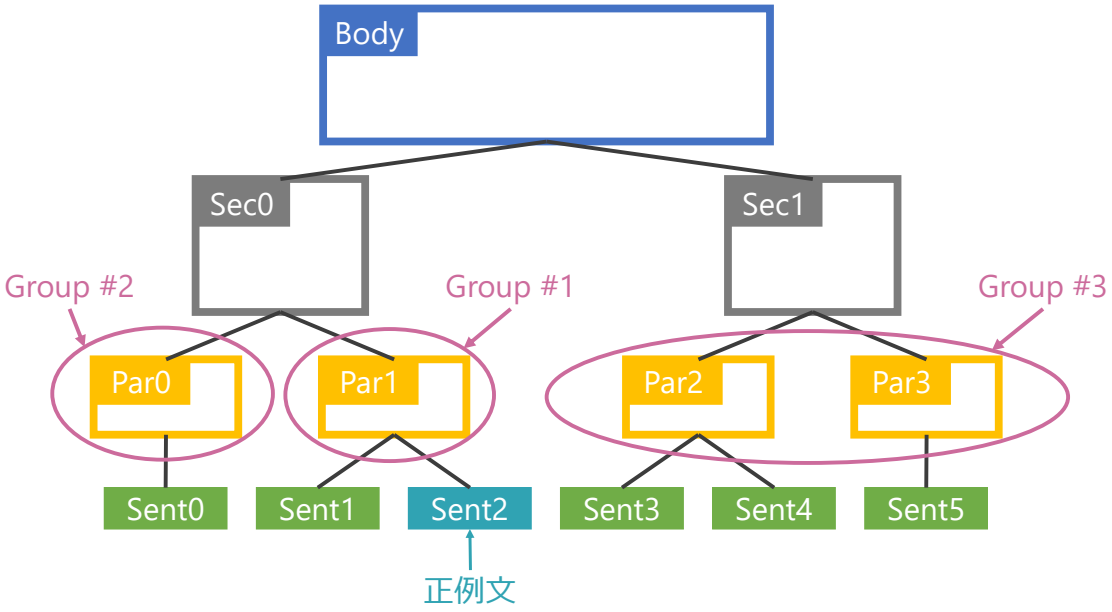


図 4.3: パラグラフのグループ分けの例. 正例文である文 2 を含んでいるパラグラフ 1 が Group #1, Group #1 と同じセクション内にあるパラグラフ 0 が Group #2, 親セクション内に正例文が一つも入っていないパラグラフ 2 およびパラグラフ 3 は Group #3 となる.

まり外れを引いてしまうが, 一口に外れの文といっても外れの度合いには差があると考えられる. なぜならば, 論文のように文書のサイズが大きい場合は, 文書内で内容的に重要な箇所と重要でない箇所が存在すると考えられ, 例え負例の文を選んでしまったとしてもそこが重要な文脈の中にあるならばそれはある程度の意味があるといえるし, 全く重要でない箇所から選んでしまえば意味は薄いといえるからである. （ただし要約というタスクを踏まえ, ここでいう重要か・重要でないかは「アブストラクトとどれだけ関連性があるか」ということを指すこととする.）直感的には正例文と同じ段落に入っている文は負例であったとしても, 正例文と同じトピックについて言及している可能性が高く, アブストラクトとの関連性が高そうである. 同様に, 正例文とセクションに入っている文も多少は関連している可能性がある. 逆に正例文が一つも入っていないセクション内の文は内容的にはあまり重要でないといえる. さきほどの提案を組み込めば, モデルは親ノードが負例の木の中の文よりも親ノードが正例の木の中の文を選ぼうとするはずであるので, 結果的にアブストラクトとの関連の低い文は除去されるのはないかと考えられる.

ここで, そもそも本当に正例文と近い文脈の方がアブストラクトとの関連性が高いのか定量的に考察するため, 以下のような予備実験を行った. まず, 論文中の全パラグラフを以下の三つのグループに分ける.

| Group | ROUGE-1 | ROUGE-2 |
|---|---|---|
| Group #1 | **33.7** | **12.2** |
| Group #1 (NEG) | 25.4 | 6.25 |
| Group #2 | 25.1 | 5.39 |
| Group #3 | 20.9 | 3.90 |

表 4.1: 各グループの ROUGE-1 および ROUGE-2 の $F_1$ スコアの平均値

**Group #1**
正例文が属しているパラグラフ

**Group #2**
自身は正例文を持たないが, 正例文を含むパラグラフ（つまり Group #1）と同じセクション内にあるパラグラフ

**Group #3**
親セクションの中に正例文が一つも含まれてないようなパラグラフ

例として, 図 4.2 に示した論文についてこのグループ分けを適用すると図 4.3 のようになる. そして, 学習データの全ての論文をパラグラフ単位で分解して, 各論文においてアブストラクトとそれぞれのパラグラフとの ROUGE スコア [21] を調査した. グループごとの ROUGE $F_1$ スコアを平均した結果を以下の表 4.1 に示す. ここで, Group #1 は正例文の寄与が大きいことも考慮し, Group #1 から正例文を除いたのが Group #1 (NEG) である. これを見ると, Group #1, Group #1 (NEG), Group #2, Group #3 の順に ROUGE スコアが高い, すなわちアブストラクトとよく単語が一致していることが分かる. 正例文はそもそもアブストラクトとバイグラムが一致している文が選ばれているので, 正例文を含んでいる Group #1 が最も高いスコアになるのはある種当然であるが, 正例文を含んでいない残りの三つのグループについてみても正例文に近い順に高いスコアを出していることに注目したい. すなわち, より正例文に近い段落の方が（たとえ自身が正例文を含んでいなかったとしても）重要な内容を述べているということが, 定量的に見てある程度の妥当性を持っていることが判明した. 先の提案に基づき, 間違いの文自体を減らすことのみならず同時に Group #3, Group #2 の間違いを少なくし, より Group #1 (NEG) に寄せていくことも目指す.

## 4.2 提案モデル

提案モデルの具体的な中身について説明する. 提案モデルの処理の流れは大まかには次のようになっている.

- 論文中の全文をそれぞれ独立にベクトル化する

- 文のベクトルの系列をパラグラフのベクトルへ, パラグラフのベクトルの系列をセクションのベクトルへ, セクションのベクトルの系列を文書のベクトルへエンコードする
- セクションのスコアを計算し, それを基に子パラグラフのスコアを計算し, さらにその子センテンスのスコアを計算する. この処理をセクションごとに繰り返す.
- 全てのスコアを計算し終えたら, スコアの高い文を順に単語数の制限を超えない範囲で抽出する

まず, 提案モデルを説明する上で使う数学的な記号および問題を定式化したのち, 各手順の具体的な説明を行う.

### 4.2.1 定式化

$x$ を論文とする. 次に $c$, $p$, $s$ および $w$ をそれぞれセクション, パラグラフ, 文および単語を表すものとし, それぞれを表すベクトルを $\boldsymbol{v}^{sec}$, $\boldsymbol{v}^{par}$, $\boldsymbol{v}^{sent}$ および $\boldsymbol{w}$ と表記することにする. また, これらの要素は必ず上から順に階層的に構成されているものする. (つまり本モデルにおいては, サブセクションも一つの独立したセクションとみなし, セクション間の包含関係は無視する. また, パラグラフがパラグラフを含んでいるような構造も考えないことにする.) さらに, ある要素中に含まれる下位要素のシーケンスの先頭を表す添え字を $top$, 末尾を表す添え字を $end$ と書くことにする. 例えば, 図 4.1 に示した論文の例では, パラグラフ 1 は文 1 と文 2 によって構成されているので, $p_1 = [s_1, s_2]$ と書くことができ, $s_{top} = s_1$, $s_{end} = s_2$ である. また, あるノードに対する親ノードの添え字を $parent$ と表記することとする.

文, パラグラフ, セクションの正解ラベルをそれぞれ $y^{sent}$, $y^{par}$, $y^{sec}$ と表記し, スコアについても同様に $p^{sent}$, $p^{par}$, $p^{sec}$ と書くことにする. 学習の過程では, 文, パラグラフおよびセクションのスコアの系列がそれぞれ正解ラベルの系列と一致するようにモデルのパラメーターを調整したいので, 提案モデルの誤差関数を以下のように設計する.

$$
\begin{aligned}
E \quad = \quad & -\sum_i \left\{ y_i^{sent} \log p_i^{sent} + (1 - y_i^{sent}) \log (1 - p_i^{sent}) \right\} \\
& -\sum_j \left\{ y_j^{par} \log p_j^{par} + (1 - y_j^{par}) \log (1 - p_j^{par}) \right\} \\
& -\sum_k \left\{ y_k^{sec} \log p_k^{sec} + (1 - y_k^{sec}) \log (1 - p_k^{sec}) \right\}
\end{aligned}
\tag{4.1}
$$

これは文, パラグラフ, セクションの三つに対してバイナリクロスエントロピーを設けたものなっている.

### 4.2.2 畳み込みニューラルネットワークを用いた文ベクトルの構成

処理のはじめとして, 畳み込みニューラルネットワーク (Convolutional Neural Network, CNN) を用いて論文の各文をベクトル化する. CNN は画像処理の分野でよく用いられるニューラルネットワー

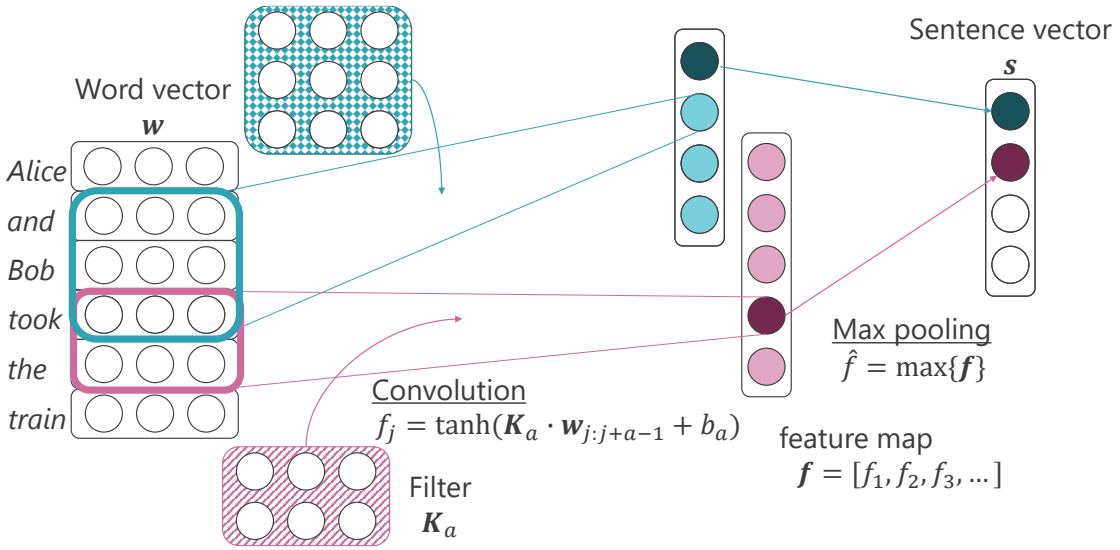


図 4.4: 提案モデルにおける CNN の構造. “Alice and Bob took the train” という文に対して単語幅 3 のサイズを持つフィルター（青色）と単語幅 2 のフィルター（赤色）が入力行列を走査している. 単語ベクトルの次元は 3, 文のベクトルの次元は 4 である.

クで [31], パターン検出を得意とする. CNN はフィルターと呼ばれる行列をパラメーターとして持っており, 入力画像中の部分画像との要素積の和を計算する. フィルターは通常数百から数千用意されており, それぞれが何らかの特徴的なパターンを持っている. 各フィルターは画像中を走査して, 部分画像との要素積を取ることで相関を調べ, 画像中のどこにその特徴的なパターンが存在するかを検出する役割がある.

言語処理の分野では, 文中に含まれる単語ベクトルを並べたものを画像とみなして CNN を用いることが多い. 画像処理とは異なり, フィルターの幅は単語ベクトルの次元と同じになっており, ある一文がある単語系列パターンに対してどれほど反応したかを検出する. 特徴的な単語の N-Gram(連続した N 個の単語) のパターンを抽出することで, 文の分類タスクなどで効果的であると考えられている [32]. 要約においても文を要約文に含めるか, 含めないかの二値分類タスクであるので CNN を利用する研究もある [9, 10]. 本研究で用いる CNN の構成は基本的には [10] と同じである. 以下, CNN の処理について説明する. ある文の中に含まれる単語ベクトルを横に並べて作った行列を

$$\boldsymbol{w}_{1:end} = \boldsymbol{w}_1 \oplus \boldsymbol{w}_2 \oplus \ldots \oplus \boldsymbol{w}_{end} \tag{4.2}$$

と表現する. ここで $\oplus$ はベクトルを行方向に連結する操作を表し, $\boldsymbol{w}_{1:end}$ が CNN に入力される. このように表現された行列を画像とみなし, CNN のフィルターが画像をスキャンする. この処理を行うパートを畳み込み層という. CNN は様々な横幅のフィルターを何枚かずつ持っており, ある横幅 $a$ の $i$ 枚目のフィルター $\boldsymbol{K}_a^{(i)}$ と, $j$ 番目の部分画像 $\boldsymbol{w}_{j:j+a-1}$ との畳み込みの値（一般に特徴量と呼ば

れる）$f_{a,j}^{(i)}$ は以下のように計算される．

$$f_{a,j}^{(i)} = \tanh(\boldsymbol{K}_a^{(i)} \otimes \boldsymbol{w}_{j:j+a-1} + b_a^{(i)}) \tag{4.3}$$

ここで, $\otimes$ は行列同士の要素積の和, $b_a^{(i)}$ はバイアス項を表す．この操作は走査できるぶんだけ行われ, 部分画像の系列 $[\boldsymbol{w}_{1:a}, \boldsymbol{w}_{2:a+1}, \cdots, \boldsymbol{w}_{end-a+1:end}]$ に対して特徴マップ

$$\boldsymbol{f}_a^{(i)} = [f_{a,1}^{(i)}, f_{a,2}^{(i)}, \cdots, f_{a,end-a+1}^{(i)}] \tag{4.4}$$

が得られる．次のプーリング層では, $i$ 番目のフィルターが画像をスイープすることによって得られた特徴マップの内, 一定の範囲ごとに代表的な値を出力する．畳み込み層ではパターン検出の位置選択性が厳密であるが, プーリング層は厳密ではなく, 入力パターンが多少ずれていたとしても反応することで, 検出の柔軟性を上げている．代表的な pooling 方法として, 範囲内での特徴量のうち最大値を出力する max pooling という方法がある．

$$\hat{f}_a^i = \max\{\boldsymbol{f}_a^{(i)}\} \tag{4.5}$$

CNN が持つすべてのフィルターについて同様の処理を行い, 得られた $\hat{f}_a^i$ を全て並べたものが文のベクトル $\boldsymbol{v}^{sent}$ となる．

$$\boldsymbol{v}^{sent} = [\hat{f}_1^1 \ ; \ \hat{f}_1^2 \ ; \ \cdots] \tag{4.6}$$

ただし, ; はベクトルを列方向に結合する処理を表すものとする．これらの処理の流れを図 4.4 に示す．

### 4.2.3 エンコーダー

エンコーダーでは CNN によって構成された文のベクトルを読み込む処理を行う．提案モデルでは, LSTM をエンコーダー・デコーダーのための RNN として用いる．提案するエンコーダー・デコーダーモデルは三層からなっており, それらをセンテンス・レイヤー, パラグラフ・レイヤー, セクション・レイヤーと呼ぶことにし, 例えばエンコーダー側のセンテンス・レイヤーの LSTM を $\mathrm{LSTM}_{enc}^{sent}$ などと表記することにする．

エンコーダー側もデコーダー側も基本的にはセクションごとに処理を行い, セクション内ではパラグラフ単位で処理を行う．まず各パラグラフ内に含まれる文について, 先頭から順にセンテンス・レイヤーの LSTM に文ベクトルを入力する．

$$\boldsymbol{h}_t^{sent} = \mathrm{LSTM}_{enc}^{sent}(\boldsymbol{v}_t^{sent}, \boldsymbol{h}_{t-1}^{sent}) \tag{4.7}$$

ただし, パラグラフの先頭においては入力として受け取る隠れ状態のベクトルはゼロベクトルとする．

$$\boldsymbol{h}_{top}^{sent} = \mathrm{LSTM}_{enc}^{sent}(\boldsymbol{v}_{top}^{sent}, \boldsymbol{0}) \tag{4.8}$$

また, パラグラフ内で最後尾の LSTM ユニットの隠れ状態ベクトルをこのパラグラフを表現するベクトルとみなす．これをパラグラフベクトルと書くことにする．

$$\boldsymbol{h}_{end}^{sent} = \mathrm{LSTM}_{enc}^{sent}(\boldsymbol{v}_{end}^{sent}, \boldsymbol{h}_{end-1}^{sent}) \tag{4.9}$$

$$\boldsymbol{v}_{parent}^{par} = \boldsymbol{h}_{end}^{sent} \tag{4.10}$$

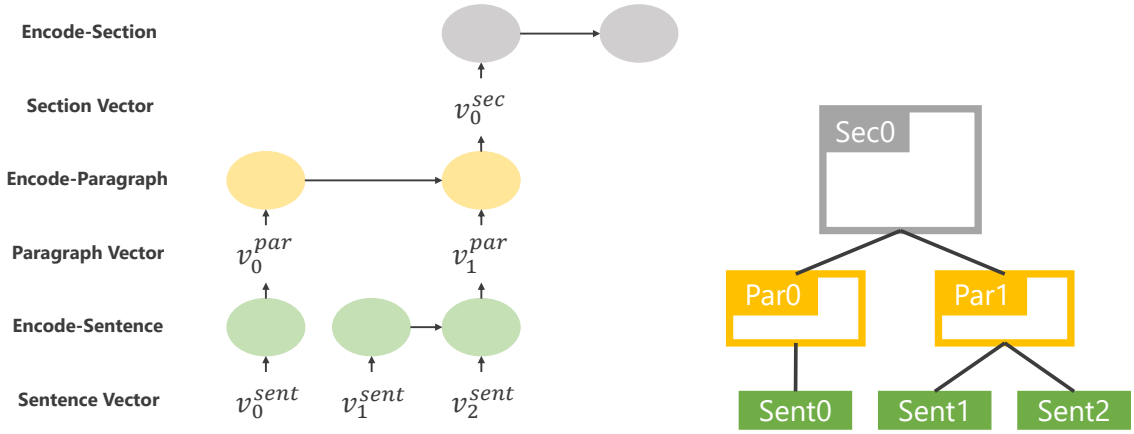


図 4.5: エンコーダーの処理の例. 緑, 黄色, 灰色のブロックはそれぞれ $\mathrm{LSTM}_{enc}^{sent}$, $\mathrm{LSTM}_{enc}^{par}$, $\mathrm{LSTM}_{enc}^{sec}$ を表す.

次に, セクション内のパラグラフについて, 先頭から順にパラグラフ・レイヤーの LSTM にパラグラフベクトルを入力する.

$$\boldsymbol{h}_t^{par} = \mathrm{LSTM}_{enc}^{par}(\boldsymbol{v}_t^{par}, \boldsymbol{h}_{t-1}^{par}) \tag{4.11}$$

この際もセンテンス・レイヤーでの処理と同様に, 先頭のパラグラフの際は一時刻前の隠れ状態のベクトルをゼロベクトルとし, 末尾のパラグラフの際はその LSTM の隠れ状態ベクトルをセクションベクトルとする.

$$\boldsymbol{h}_{top}^{par} = \mathrm{LSTM}_{enc}^{par}(\boldsymbol{v}_{top}^{par}, \boldsymbol{0}) \tag{4.12}$$

$$\boldsymbol{h}_{end}^{par} = \mathrm{LSTM}_{enc}^{par}(\boldsymbol{v}_{end}^{par}, \boldsymbol{h}_{end-1}^{par}) \tag{4.13}$$

$$\boldsymbol{v}_{parent}^{sec} = \boldsymbol{h}_{end}^{par} \tag{4.14}$$

最後にセクション・レイヤーでも同様の処理を行う.

$$\boldsymbol{h}_{top}^{sec} = \mathrm{LSTM}_{enc}^{sec}(\boldsymbol{v}_{top}^{sec}, \boldsymbol{0}) \tag{4.15}$$

$$\boldsymbol{h}_t^{sec} = \mathrm{LSTM}_{enc}^{sec}(\boldsymbol{v}_t^{sec}, \boldsymbol{h}_{t-1}^{sec}) \tag{4.16}$$

$$\boldsymbol{h}_{end}^{sec} = \mathrm{LSTM}_{enc}^{sec}(\boldsymbol{v}_{end}^{sec}, \boldsymbol{h}_{end-1}^{sec}) \tag{4.17}$$

これらの処理の流れを図 4.5 に示す.

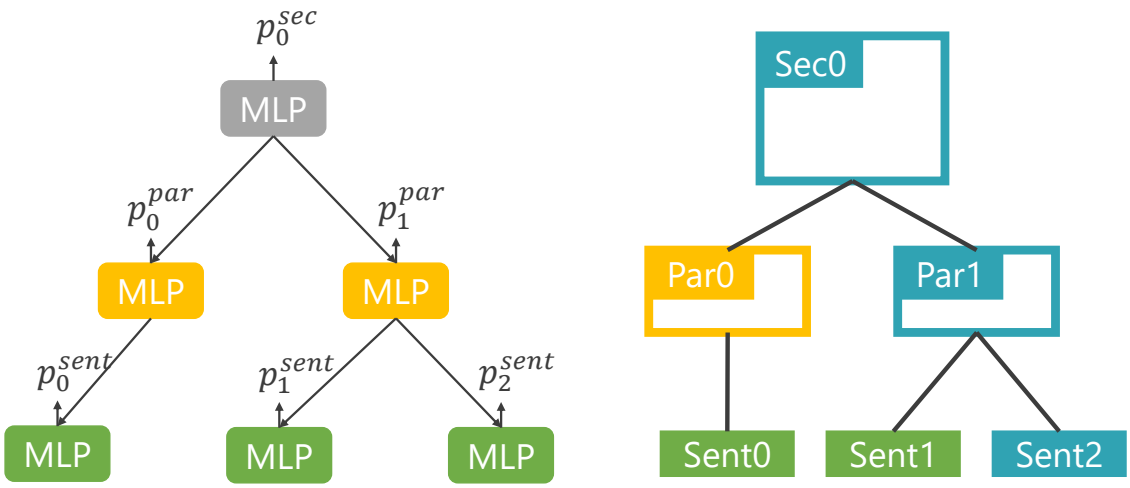


図 4.6: 木構造に基づく階層的なスコア計算の例.

### 4.2.4 デコーダー

次にデコーダー側の処理について説明する. デコーダー側は系列情報を伝搬するための三層の LSTM と, セクション, パラグラフ, センテンスそれぞれのスコアを計算するための三つの MLP によって構成されている. 分類器である MLP は, そのレイヤーのデコーダーの隠れ状態ベクトルとそれに対応するエンコーダーの隠れ状態ベクトル, そして親ノードの情報を入力として受け取る. 定性的な意味付けとしては, エンコーダーの隠れ状態ベクトルはそのノード自体の意味情報を表現し, デコーダーの隠れ状態ベクトルはそれまでの系列がどれくらい要約として含まれていたかという履歴の情報を持っていると考えられる.

ここで, 最終的に欲しいのはセンテンスのスコアのみだが数百文の中から正確に正例を予測するのは難しいので, 木構造を活かした階層的なスコア計算を行う. 正例文の親パラグラフ・親セクションは必ず正例になるように定義しているので, 基本的にはモデルはスコアの高いノードをたどっていけば正例文にたどり着きやすくなる. いきなりセンテンスのスコアのみを計算させるのではなく, 親ノードから子ノードへ順にスコアを計算し, 最終的にモデルがより正解の文に高いスコアを付与できるようサポートするということである. そのために親ノードのスコアの情報を子ノードに伝搬させることを考える. 具体的にノード間を伝搬させる情報として, 各 MLP 中の隠れ状態ベクトルを使う. MLP 中の隠れ状態ベクトルはそのノードのスコアの計算に直接使われ, さらにそのスコアは誤差関数に組み込まれているため, 誤差逆伝搬によって隠れ状態ベクトルもそのノードの自身がどれほど重要であるかという特徴量を含有するように学習されるからである. この様子を図 4.6 に示す.

エンコーダーではセンテンス, パラグラフ, セクションの順にベクトルを伝搬したが, デコーダーではそれとは逆順に処理を行う. まず, エンコーダー側からデコーダー側に処理が移るとき, まず初めに

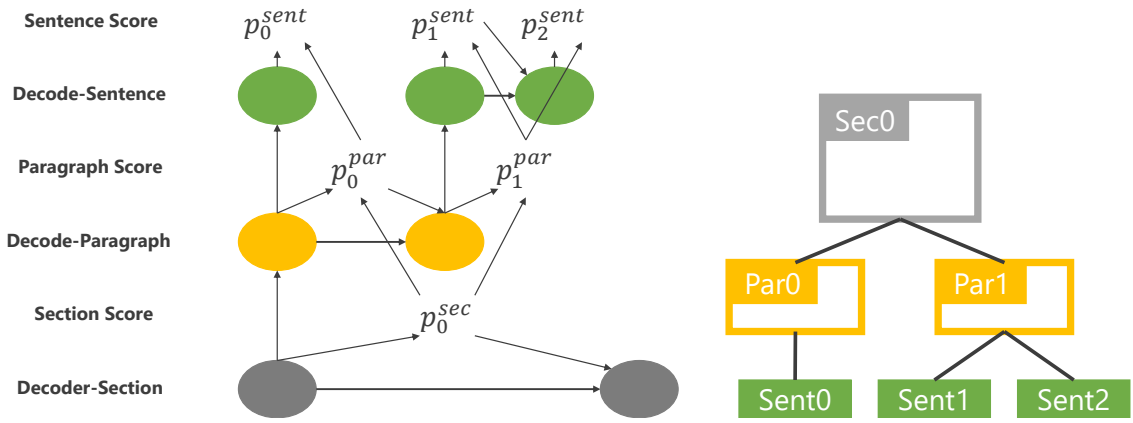


図 4.7: デコーダーの処理の例. 緑, 黄色, 灰色のブロックはそれぞれ $\mathrm{LSTM}_{dec}^{sent}$, $\mathrm{LSTM}_{dec}^{par}$, $\mathrm{LSTM}_{dec}^{sec}$ を表す.

エンコーダー側の末尾のセクション・レイヤーの LSTM ユニットの隠れ状態ベクトルをデコーダー側の先頭のセクション・レイヤーの LSTM ユニットに渡す. このとき, 入力は文書の終わりを表すベクトル (End Of Document, EOD) とする. EOD ベクトルもモデルのパラメーターの一部であり, 学習によるアップデートの対象である.

$$\bar{\boldsymbol{h}}_{top}^{sec} = \mathrm{LSTM}_{dec}^{sec}(\mathrm{EOD}, \boldsymbol{h}_{end}^{sec}) \tag{4.18}$$

セクション・レイヤーでは, セクションのスコアの系列を予測する. LSTM には一時刻前のセクションベクトルとスコアをかけたものを入力する.

$$\bar{\boldsymbol{h}}_{t}^{sec} = \mathrm{LSTM}_{dec}^{sec}(p_{t-1}^{sec}\boldsymbol{v}_{t-1}^{sec}, \bar{\boldsymbol{h}}_{t-1}^{sec}) \tag{4.19}$$

スコアの算出はデコーダー側の隠れ状態ベクトルとそれに対応するエンコーダー側の隠れ状態ベクトルを入力として, 三層の MLP によって行う.

$$\tilde{\boldsymbol{h}}_{t}^{sec} = \tanh(W_1^{sec}[\boldsymbol{h}_{t}^{sec}; \bar{\boldsymbol{h}}_{t}^{sec}] + \boldsymbol{b}_1^{sec}) \tag{4.20}$$

$$p_{t}^{sec} = \sigma(W_2^{sec}\tilde{\boldsymbol{h}}_{t}^{sec} + b_2^{sec}) \tag{4.21}$$

なお, $\tilde{\boldsymbol{h}}_{t}^{sec}$ はこの MLP における隠れ状態ベクトルである.

パラグラフ・レイヤーでは先頭の LSTM ユニットがまず親セクションの隠れ状態ベクトルを入力に受け取る.

$$\bar{\boldsymbol{h}}_{top}^{par} = \mathrm{LSTM}_{dec}^{par}(\bar{\boldsymbol{h}}_{parent}^{sec}, \boldsymbol{0}) \tag{4.22}$$

先頭以降では, 一時刻前のパラグラフベクトルとそのスコアをかけたものを入力する.

$$\bar{\boldsymbol{h}}_{t}^{par} = \mathrm{LSTM}_{dec}^{par}(p_{t-1}^{par}\boldsymbol{v}_{t-1}^{par}, \bar{\boldsymbol{h}}_{t-1}^{par}) \tag{4.23}$$

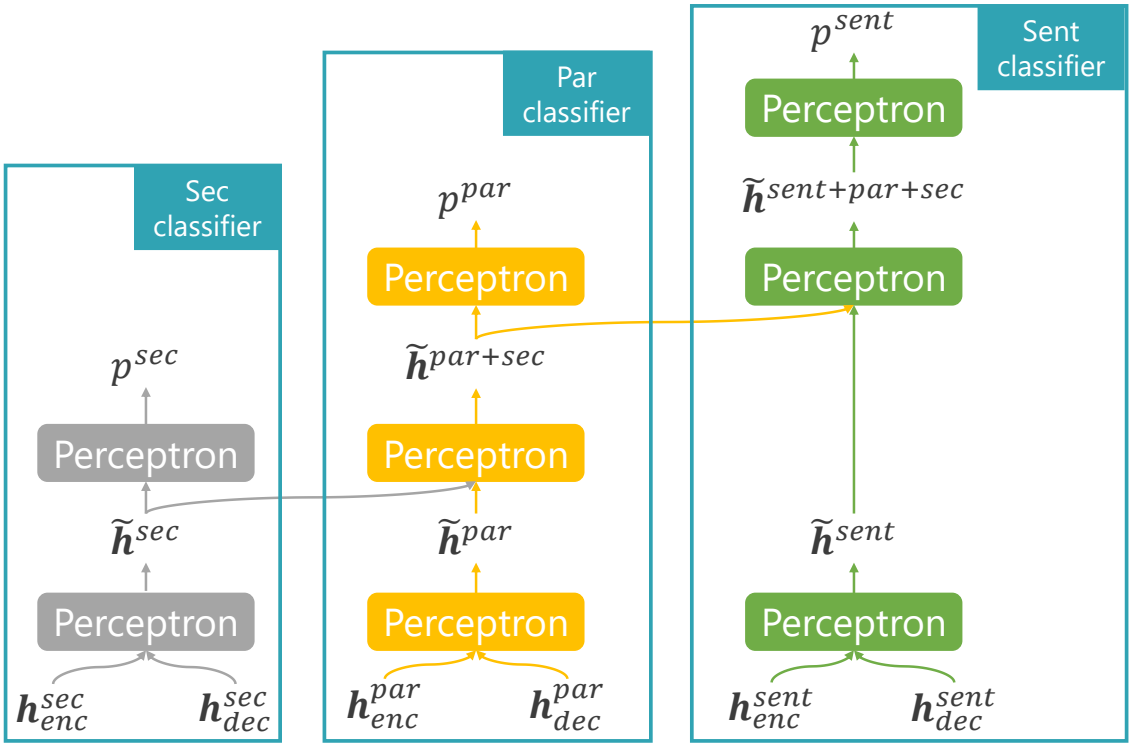


図 4.8: 親子関係を考慮した分類器.

スコアの算出はデコーダーの隠れ状態ベクトルとそれに対応するエンコーダーの隠れ状態ベクトルを入力として,MLP によって行う. ただし, 隠れ層を一層多くして途中から親セクションの MLP の隠れ状態ベクトル $\tilde{\boldsymbol{h}}^{sec}_{parent}$ を入力として加える.

$$\tilde{\boldsymbol{h}}^{par}_t = \tanh(W^{par}_1[\boldsymbol{h}^{par}_t; \bar{\boldsymbol{h}}^{par}_t] + \boldsymbol{b}^{par}_1) \tag{4.24}$$

$$\tilde{\boldsymbol{h}}^{par+sec}_t = \tanh(W^{par}_2[\tilde{\boldsymbol{h}}^{sec}_{parent}; \tilde{\boldsymbol{h}}^{par}_t] + \boldsymbol{b}^{par}_2) \tag{4.25}$$

$$p^{par}_t = \sigma(W^{par}_3 \tilde{\boldsymbol{h}}^{par+sec}_t + b^{par}_3) \tag{4.26}$$

なお, $\tilde{\boldsymbol{h}}^{par+sec}_t$ はこの MLP における隠れ状態ベクトルで, このパラグラフとその親セクションの情報を含んでいる.

センテンス・レイヤーでは, 先頭の LSTM ユニットがまず親パラグラフの隠れ状態ベクトルを入力に受け取る.

$$\bar{\boldsymbol{h}}^{sent}_{top} = \mathrm{LSTM}^{sent}_{dec}(\bar{\boldsymbol{h}}^{par}_{parent}, \boldsymbol{0}) \tag{4.27}$$

先頭以降では, 一時刻前のセンテンスベクトルとそのスコアをかけたものを入力する.

$$\bar{\boldsymbol{h}}_t^{sent} = \mathrm{LSTM}_{dec}^{sent}(p_{t-1}^{sent}\boldsymbol{v}_{t-1}^{sent}, \bar{\boldsymbol{h}}_{t-1}^{sent}) \tag{4.28}$$

スコアの算出はデコーダーの隠れ状態ベクトルとそれに対応するエンコーダーの隠れ状態ベクトルを入力として, MLP によって行う. ただし, 隠れ層を一層多くして途中から親セクションおよび親パラグラフの情報を含んだ MLP の隠れ状態ベクトル $\tilde{\boldsymbol{h}}_{parent}^{par+sec}$ を入力として加える.

$$\tilde{\boldsymbol{h}}_t^{sent} = \tanh(W_1^{sent}[\boldsymbol{h}_t^{par}; \bar{\boldsymbol{h}}_t^{par}] + \boldsymbol{b}_1^{sent}) \tag{4.29}$$

$$\tilde{\boldsymbol{h}}_t^{sent+par+sec} = \tanh(W_2^{sent}[\tilde{\boldsymbol{h}}_{parent}^{par+sec}; \tilde{\boldsymbol{h}}_t^{sent}] + \boldsymbol{b}_2^{sent}) \tag{4.30}$$

$$p_t^{sent} = \sigma(W_3^{sent}\tilde{\boldsymbol{h}}_t^{sent+par+sec} + b_3^{sent}) \tag{4.31}$$

なお, $\tilde{\boldsymbol{h}}_t^{sent+par+sec}$ はこの MLP における隠れ状態ベクトルで, このセンテンスとその親パラグラフ・親セクションの情報を含んでいる. 図 4.8 に三つの MLP によるスコア計算の処理の様子を示す.

なお学習時はいずれのレイヤーでも, 機械翻訳や Cheng と Lapata のモデルと同様, これまでの時刻の正解ラベルの系列を所与として LSTM の計算を行う. よって, デコーダーの LSTM の計算は以下のようになる.

$$\bar{\boldsymbol{h}}_t^{sec} = \begin{cases} \mathrm{LSTM}_{dec}^{sec}(\boldsymbol{0}, \bar{\boldsymbol{h}}_{t-1}^{sec}) & \text{if } \ y_{t-1}^{sec} = 0 \\ \mathrm{LSTM}_{dec}^{sec}(\boldsymbol{v}_{t-1}^{sec}, \bar{\boldsymbol{h}}_{t-1}^{sec}) & \text{if } \ y_{t-1}^{sec} = 1 \end{cases} \tag{4.32}$$

$$\bar{\boldsymbol{h}}_t^{par} = \begin{cases} \mathrm{LSTM}_{dec}^{par}(\boldsymbol{0}, \bar{\boldsymbol{h}}_{t-1}^{par}) & \text{if } \ y_{t-1}^{par} = 0 \\ \mathrm{LSTM}_{dec}^{par}(\boldsymbol{v}_{t-1}^{par}, \bar{\boldsymbol{h}}_{t-1}^{par}) & \text{if } \ y_{t-1}^{par} = 1 \end{cases} \tag{4.33}$$

$$\bar{\boldsymbol{h}}_t^{sent} = \begin{cases} \mathrm{LSTM}_{dec}^{sent}(\boldsymbol{0}, \bar{\boldsymbol{h}}_{t-1}^{sent}) & \text{if } \ y_{t-1}^{sent} = 0 \\ \mathrm{LSTM}_{dec}^{sent}(\boldsymbol{v}_{t-1}^{sent}, \bar{\boldsymbol{h}}_{t-1}^{sent}) & \text{if } \ y_{t-1}^{sent} = 1 \end{cases} \tag{4.34}$$

のように学習する. これらの処理の流れを図 4.7 に示す.

# 第5章　実験

本章では提案手法の性能を測るために行った比較実験について報告する．まず, 学術論文を要約のデータセットとして用いるために行った前処理について述べ, 次に具体的な実験の結果と考察について述べる．

## 5.1　データセット

データセットとして, 主要医学系雑誌等に掲載された論文のデータベースである PubMed Central[1] が提供している論文を用いる．このデータベースからは xml 形式で生命医学系の論文がダウンロードできる．まず, セクションやパラグラフなど文書構造を表すタグ以外の不要なタグ, 参考文献, 図や表をすべて取り除き, タイトル, キーワード, 本文とアブストラクトのみをテキストデータとして扱えるようにする．

次に, Stanford coreNLP[2] を用いてトークン化および文分割をおこなう．トークン化とはテキスト中の語に適切に区切りを入れることである．英語はもともと単語と単語の間に空白が入り自動で区別されている言語であるが, 単語と記号が接着して書かれている場合は切り離す必要がある．例えば“This is a pen.”という文の中の“pen”と“.”は切り離されなくてはならない．文分割とは一続きになっているテキストを文単位で区切ることである．本研究が想定する抽出型要約では文単位で抽出を行うため, 前処理としてテキストを適切に文単位に区切っておく必要がある．日本語と違い, 英語ではピリオドが文末の単語だけではなく短縮形の単語にも付随するため (“Mr.”など), 100% の精度で文を分割するのは難しい．実際 Stanford coreNLP の文分割処理を上記のデータセットにそのまま適用したところ誤りが多かったため, ピリオドを含む単語で Stanford coreNLP が文分割のタイミ

| | Train | Valid | Test |
|---|---|---|---|
| Section | 15.0 | 15.1 | 15.1 |
| Paragraph | 36.0 | 36.0 | 35.8 |
| Sentence | 164.0 | 165.6 | 163.0 |
| Word | 4507.2 | 4551.1 | 4494.1 |

表 5.1: データセット中の各論文に含まれるセクション数, パラグラフ数, 文数および単語数の平均．

[1] ftp://ftp.ncbi.nlm.nih.gov/pub/pmc/oa_bulk/
[2] http://stanfordnlp.github.io/CoreNLP/

**Background:**
It is well known that the organization of genes within eukaryotic genomes is non-random [1, 2, 3]. The functionally linked genes may occur in either loose groups (i.e. they are not necessarily located in immediate vicinity but enriched in several areas), or tightly packed clusters such as those involved in secondary metabolite synthesis in filamentous fungi [4] or the small secreted effector proteins in plant pathogenic fungi [5]. Regions that contain the most actively expressed genes have a higher gene density [6, 7] and a high G/C content [7]. Numerous genome-wide gene expression analyses revealed that a large portion of co-expressed genes are also co-localized in specific areas of chromosomes [8, 9, 10, 11]. (...)

**XML:**

<sec sec-type="background" id="0">

<s id="0"> It is well known that the organization of genes within eukaryotic genomes is non-random -LSB- 1 – 3 -RSB- . </s >
<s id="1"> The functionally linked genes may occur in either loose groups -LRB- i.e. they are not necessarily located in immediate vicinity but enriched in several areas -RRB- , or tightly packed clusters such as those involved in secondary metabolite synthesis in filamentous fungi -LSB- 4 -RSB- or the small secreted effector proteins in plant pathogenic fungi -LSB- 5 -RSB- . </s >
<s id="2"> Regions that contain the most actively expressed genes have a higher gene density -LSB- 6 , 7 -RSB- and a high G/C content -LSB- 7 -RSB- . </s >
<s id="3"> Numerous genome-wide gene expression analyses revealed that a large portion of co-expressed genes are also co-localized in specific areas of chromosomes -LSB- 8 – 11 -RSB- . </s >
 (...)

図 5.1: xml 形式で記述された論文の例. 上段がある論文中の Background セクション中の一部, 下段がそれに対応するトークン化および文分割を施し, 文書構造をタグで管理している. sec がセクション, p がパラグラフ, s が文を表すタグである.

ングを間違える単語を可能な限りリストアップし, それらを一度ダミーの単語に置き換え, Stanford coreNLP で文分割処理を行った後, ダミーの単語を元に戻した. これらの処理後に, 各 xml ファイルに新たに文の始まりと終わりを示すタグを挿入し, 文単位で情報を扱えるようにする.

このようにしてできたデータセットのうち「アブストラクトが 75 単語以上」かつ「本文の文の数が 300 以下」かつ「一文に含まれる単語数が 100 単語以内」の三つの条件を満たすものの中から 30,000 本を実験に用いる. これらの 90%を学習データ, 5%を開発データ, 5%をテストデータとする. データごとのサイズに関する分布を表 5.1 に示す. また, 実験においては, 各単語をすべて小文字化し, 学習データ中の登場回数が 5 回未満の単語は全て未知語として“ UNK ”という特殊なトークンに置換して処理する. 最後に, 本実験で用いたデータセットはサンプルによってサイズが異なるので, 出力する要約の長さをその論文の本文の単語数の 6%内に制限することにする. 実際に実験で用いたデータの例を図 5.1 に示す.

## 5.2 評価手法

### 5.2.1 ROUGE スコア

多くの要約の研究では ROUGE という指標 [21] が評価手法として用いられることが多い. これは参照要約とシステム要約の N-Gram(連続した N 個並びの単語) がどれだけマッチしているかを見る指標である.

$$\text{ROUGE-N}_{\text{Precision}} = \frac{\sum_{g \in R} \min\left(F_R(g), F_S(g)\right)}{\sum_{g \in R} F_R(g)} \tag{5.1}$$

$$\text{ROUGE-N}_{\text{Recall}} = \frac{\sum_{g \in R} \min\left(F_R(g), F_S(g)\right)}{\sum_{g \in S} F_S(g)} \tag{5.2}$$

$$\text{ROUGE-N}_{\text{F}} = \frac{2 \cdot \text{ROUGE-N}_{\text{Precision}} \cdot \text{ROUGE-N}_{\text{Recall}}}{\text{ROUGE-N}_{\text{Precision}} + \text{ROUGE-N}_{\text{Recall}}} \tag{5.3}$$

ただし, R, S はそれぞれ参照要約, システム要約を表し, $F_R(g)$, $F_S(g)$ はある N-Gram $g$ についてそれぞれの要約に含まれている数を返す関数とする. ROUGE は著者がツールキットとして実装し, 公開されている [3]. また, ツールを使用する際には stemming する, stop word は除去するなどさまざまなオプションを設定できるが, 本実験では使用するオプションは先行研究に合わせた [4].

一方で, 近年では ROUGE による評価と人間による評価との間の剥離を指摘する研究や, ROUGE を改良するといった研究も見受けられ, この分野の大きな課題となっている. 自動で定量評価するだけでなく, 複数の人間で評価を行う研究もある. その場合は informative などの項目を設ける.

ROUGE にはこのほかにもさまざまな指標があるが, 本研究では要約の研究でよく用いられている ROUGE-1, ROUGE-2, ROUGE-L を評価指標として用いた. また, 先に述べたようにテストデータとして用いる論文はサイズ（中に含まれている単語数）が様々なので, 上記三つの F 値を採用した.

### 5.2.2 文番号集合の被覆度

ROUGE スコアはあくまでシステムが選び出した文の集合とアブストラクトとの N-Gram の被覆度合を測る指標であるため, 仮に正例以外の文を選んでもその文がアブストラクトと共通の N-Gram を持っていればスコアに寄与する可能性がある. したがって, モデルが教師信号のパターンをうまく学習できたかという度合と必ずしも合致しているわけではない. そのため, それを測る指標として文番号集合の被覆度という指標を導入する. これは教師信号として事前に与えられた文の集合, つまりモデルが選ぶことが期待されている文の集合のうちいくつを拾えているかという指標である. 図 5.2 に例を示す. この例では文#2, #6, #7, #8 の四つを拾うべきところを, システムは実際には#7, #8 の二つしか拾えていない. よってこの場合の被覆度は $\frac{2}{4} = 50\%$ となる.

[3] https://github.com/summanlp/evaluation/tree/master/ROUGE-RELEASE-1.5.5
[4] -m -n 2 -w 1.2

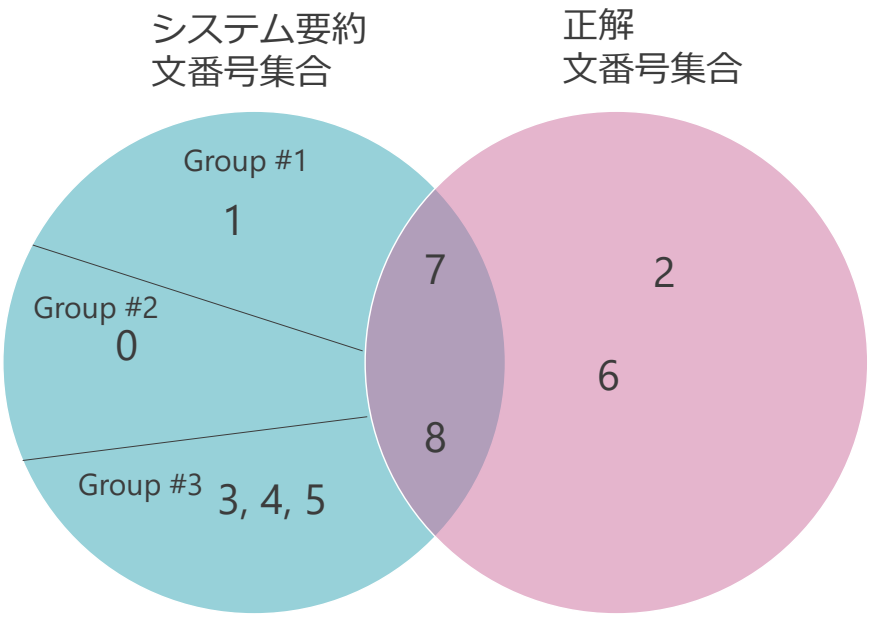


図 5.2: システム要約文集合と正解文集合の例. 中の数字は文の番号を表す.

### 5.2.3 誤りパターンの分布

先に述べたように誤りにはパターンがあり, Group #3 のような正例文から遠い箇所はアブストラクトとは関連性が薄い文脈であることが予想される. そのため, システムが間違えて選んだ文がどの文脈に属しがちかという分布を評価する. 図 5.2 に例を示す. この場合システムが間違えて選んでしまった文は#0, #1, #3, #4, #5 の五つである. このうち#1 の一つが Group #1, #0 の一つが Group #2, そして#3, #4, #5 の三つが Group #3 に属しているとすると, 誤りの分布は Group #1 $= \frac{1}{5} = 20\%$, Group #2 $= \frac{1}{5} = 20\%$, Group #3 $= \frac{3}{5} = 60\%$ ということになる. 先に述べた分析に基づき, 誤りのパターンが Group #3 よりも Group #1 に分布していること, つまりアブストラクトで述べられているのとは無関係な文の割合がそれほど低いのかを評価する. また, 割合だけでなく実際の個数も評価する.

## 5.3 教師信号

先に述べたように, 抽出型要約における正解ラベルとは正例が「要約として含まれるべき」ことを表し, 負例が「要約として含まれない」ことを表しており, そのような二値のラベルを学習データ中の全ての本文に対して設ける必要がある. 理想的には人間が正例・負例ラベルを張った学習データが

**Algorithm 1** 正解文集合の探索アルゴリズム

$BEST_{i,j}(1 \leq i \leq M, 1 \leq j \leq W) \Leftarrow \emptyset$
**for** $i = 1$ to $M$ **do**
  **for** $j = 0$ to $W$ **do**
    $BEST_{i,j} \Leftarrow BEST_{i-1,j}$
    **if** $|s_i| \leq j$ & $|ABS \cap BEST_{i,j}| < |ABS \cap BEST_{i-1,j-|s_i|} + s_i|$ **then**
      $BEST_{i,j} \Leftarrow BEST_{i-1,j-|s_i|} + s_i$
    **end if**
  **end for**
**end for**

望ましいが, そのようなデータは非常に限られており, かつ, 大規模データに人手でアノテーションを作ることは非常にコストが大きい. そのため, 機械学習ベースの抽出型要約の研究の多くは, 参照要約に対して N-gram の被覆がなるべく大きくなるような文集合を探索して, その文集合を便宜的に正解ラベルとみなしたり [5, 10], いくつかの人手で作った特徴量をもとに参照要約と類似度の高い文を本文中から選ぶための分類器を, 文を抽出する機械学習モデルとはまた別に学習し, それを用いて正解文集合を探索するなどして教師データを生成している [9].

当然ながら, 上記の手法で作った学習データは人間が用意した場合と比べてギャップがあり, 仮にモデルが十分に学習できたとしても出力要約の質が悪くなる可能性が大いにある. 特に, 文と文とのつながりや文脈が十分に考慮されておらず, 単語のマッチング度合は大きくても, 抜き出してきた文集合をつなげて読んでみたときにきちんとした要約になっている保証がない. また, そもそもこれらの手法はあらかじめ参照要約が付与されていないと適用できないという問題もある.

これらを踏まえて近年は, 教師信号として前処理で正解ラベルを作成するのではなく, あえて別のタスクと接続して end-to-end に抽出型要約のタスクを行う研究もおこなわれている [5, 10]. 例えば, Nallapati ら [5] は抽出型要約の学習に人間の書いた参照要約を直接用いる手法を提案している. また, Isonuma ら [10] は文書分類との同時学習により要約用の教師信号は使わずに単一文書の抽出型要約を行うモデルを提案している.

あるいは N-gram 被覆度ベースの探索の質を高める研究も存在するが, 依然として有効な解決策はまだ見つかっておらず, このタスクの大きな課題となっている.

本研究では N-gram 被覆度ベースの探索法, 具体的には中須賀と鶴岡 [33] が考案した動的計画法に基づく探索を使って学習データを作る. アルゴリズムの疑似コードを Algorithm 1 に示す. このアルゴリズムの目的は単語数の制約条件 (その論文の本文の 6%以下) の下で最もアブストラクトと類似する文集合を探索することである. ここでは, 類似度を表す指標としてバイグラムがどれだけ被覆したかを用いる.

次に, いくつかの数学的記号を定義する. $ABS$ と $SUM$ をそれぞれその論文のアブストラクトと, 抽出された文集合のバイグラム集合を表すとする. また, $i$ を文のインデックス, $j$ を抽出した文集合の合計単語数とする. $M$ をその論文中の本文の数, $W$ を要約の単語数の上限とし, 図 5.3 に示される

| | | Number of words included in the sentence set | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | ... | $j$ | ... | $W$ |
| Sentence IDs | 0 | | | | | | | |
| | 1 | | | | | | | |
| | 2 | | | | | | | |
| | ... | | | | | | | |
| | $i$ | | | | | $Best_{i,j}$ | | |
| | ... | | | | | | | |
| | $M$ | | | | | | | |

図 5.3: 動的計画法で用いるテーブル.

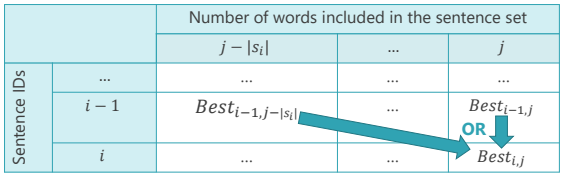


図 5.4: $Best_{i,j}$ の決定.

ようなテーブルを用意する. ここで $SUM$ と $ABS$ の ROUGE-2 の F 値は以下のように計算される.

$$\text{ROUGE} - 2\ \text{F}_1 = \frac{2 \times |\text{ABS} \cap \text{SUM}|}{|\text{ABS}| + |\text{SUM}|} \tag{5.4}$$

上式に従うと, 二つの文集合 $SUM_1$ と $SUM_2$ において, $|SUM_1| = |SUM_2|$ ならば, アブストラクトとの積集合が大きい方がスコアは大きくなる. このアルゴリズムのアイデアは各 $j$ に対して疑似的に最適な文集合を求めることである. ここで, $BEST_{i,j}$ を, 1 番目から $i$ 番目までの文の中で, 最もアブストラクトとの類似度が高く, かつ合計単語数が $j$ となるような集合を表すものとする. $BEST_{i,j}$ を決める際には二通りの場合が考えられる. 一つ目は, $BEST_{i-1,j-|s_i|}$ に $i$ 番目の文を追加する場合である. 二つ目は, $BEST_{i-1,j}$ をそのまま $BEST_{i,j}$ とする場合である. 図 5.4 に示されるように, 二つの候補のうちより類似度が高い方が $BEST_{i,j}$ となる. このアルゴリズムは $i$ の昇順でまず小さなサイズの $BEST_{i,j}$ を見つけたあと, すでに得られた結果を基により大きなサイズの $BEST_{i,j}$ を求めるという仕組みになっている. テーブルの全てを走査した後に得られる文集合を正例（つまり要約に含まれるべき）とし, それ以外の全ての文を負例（つまり要約に含まれないべき）とラベル付けする. この手法によって作成された教師信号の例を図 5.5 に示す.

**Abstract:**
Acculturation is for indigenous peoples related to the process of colonisation over centuries as well as the on-going social transition experienced in the Arctic today. Changing living conditions and lifestyle affect health in numerous ways in Arctic indigenous populations. Self-rated health (SRH) is a relevant variable in primary health care and in general public health assessments and monitoring. Exploring the relationship between acculturation and SRH in indigenous populations having experienced great societal and cultural change is thus of great importance. (...)

---

**Body:**
Acculturation is for indigenous peoples [1] related to the process of colonisation over centuries [2]. Being one of the most cited definitions [3], Redfield, Linton and Herskovits [4] define acculturation as “those phenomena which result when groups of individuals having different cultures come into continuous first-hand contact, with subsequent changes in the original culture patterns of either or both groups” (p. 149). (...)
In sum, SRH conceptually functions as a composite measure of mental and physical health [40], and becomes thus a relevant variable in primary health care and in general public health assessments and monitoring [44]. Exploring the relationship between acculturation and SRH in indigenous populations having experienced great societal and cultural change is thus of great importance. (...)

図 5.5: アブストラクトと本文中の正例文集合の例. それぞれ一部分を表示した. ただしパラグラフやセクションは無視して文のみ記載してある. 本文中の赤文字で書かれている文が正例となり, それ以外の文は全て負例となる.

## 5.4 学習の設定

単語ベクトルの次元を 300, LSTM の入力次元および出力次元を 600 とした. CNN のフィルターは幅を $a = \{1, 2, 3, 4, 5, 6\}$ とし, それぞれの幅のフィルターを 100 枚ずつ用意して文ベクトルの次元が合計で 600 次元になるようにする. MLP の隠れ層の次元はすべて 1200 とした.

モデルのパラメーターは [-0.05, 0.05] を範囲とする一様分布からの乱数に従って初期化を行った. ただし, バイアス項は $\mathbf{0}$ で初期化し, LSTM のバイアス項の一つである $\boldsymbol{bf}$ のみ $\mathbf{1}$ で初期化した [34]. また, 単語ベクトルは学習データの本文に対して word2vec [5]を用いて skip-gram モデル [25] で初期化した. 最適化は Adam [20] を用い, ハイパーパラメーターは $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1.0 \times 10^{-8}$ とした. 正則化として LSTM の入力方向と MLP にドロップアウト [35] を確率 0.5 で適用した. ミニバッチサイズは 30 とした. 学習は開発データに対する ROUGE-2 スコアを測定し,

[5] `https://github.com/tmikolov/word2vec`. 実行時のオプションは-cbow 0 -window 6 -negative 10 -hs 1 -min-count 5 -binary 0 とした.

それの値に応じて早期終了する. 実装は行列演算ライブラリ Eigen [6]を用いて C++で行い, 実験はマルチコア CPU で行った.

## 5.5 比較手法

比較手法は次のとおりである.

**LEAD**

抽出型要約の研究でよく比較手法として用いられる最も単純な手法で, 制限長に達するまで文頭から順に抜粋し要約とする. これを LEAD-SENT とする. また, 学術論文の場合はあらかじめ章構造のフォーマットが決まっていることを踏まえ, 各パラグラフの先頭の文を順に抽出する LEAD-PAR, 各セクションの先頭の文を順に抽出する LEAD-SEC も比較した.

**LREG**

ロジスティック回帰モデルに基づく教師有り学習の手法. 使用した特徴量は, 以下の通りである.

- 文中に含まれている単語
- 論文中における文の位置
- 段落中における文の位置
- セクション中における文の位置
- 論文中における, 文が属している段落の位置
- セクション中のおける, 文が属している段落の位置
- 論文中における, 文が属しているセクションの位置

**NN-SE**

Cheng と Lapata のモデル [9].

**THREE-LAYER NN-SE**

Cheng と Lapata のモデル [9] のエンコーダーとデコーダーをそれぞれ三層にしたモデル. 提案手法がそれぞれ三層の LSTM を用いているため, パラメーター数のオーダーを合わせるために設けた.

**PROPOSED**

本研究の提案手法. ただし, NN-SE, THREE-LAYER NN-SE および提案モデルにおいて文ベクトルを獲得するための CNN は共通のモデルを用いることとする.

**GOLD**

手法ではないが, 5.3 節で記載した動的計画法によって探索された正例文の集合. つまり文抽出型要約によって達成しうる疑似的な上限値を表す.

[6] `http://eigen.tuxfamily.org/index.php`

| Model | ROUGE-1 | ROUGE-2 | ROUGE-L |
|---|---|---|---|
| GOLD | 62.4 | 39.0 | 59.5 |
| LEAD-SENT | 37.1 | 10.2 | 34.3 |
| LEAD-PAR | 41.9 | 13.5 | 38.7 |
| LEAD-SEC | 41.4 | 13.2 | 38.3 |
| LREG | 49.7 | 20.6 | 46.4 |
| NN-SE | 50.0 | 21.0 | 47.9 |
| THREE-LAYER NN-SE | 50.1 | 21.3 | 48.1 |
| PROPOSED MODEL | **50.7** | **22.1** | **48.6** |

表 5.2: 各モデルが出力したシステム要約の ROUGE スコア (%).

| Model | Coverage |
|---|---|
| NN-SE | 28.9 |
| THREE-LAYER NN-SE | 30.0 |
| PROPOSED MODEL | **31.2** |

表 5.3: 文集合の被覆度 (%).

## 5.6 結果

各モデルの ROUGE スコアの値を表 5.2 に示す. これを見ると, GOLD を除いたなかでは PROPOSED MODEL が最も高い値を達成した. 次に教師有り学習ベースの手法の文集合の被覆度を表 5.3 に示す. これを見ると, PROPOSED MODEL が最も高い被覆度を記録していることが分かる. とはいえ, 正解文集合の 31%程度しか回収できておらず, このモデルは依然として改善の余地があることがわかる. さらに, THREE-LAYER NN-SE と NN-SE の違いを見ても, エンコーダーおよびデコーダーの層をそれぞれ三倍にしたにも関わらずわずかしか上がっていない. このタスクは単にパラメーター数を増やして表現能力を上げても解決には繋がらないことを示唆している.

| Model | Group #1 | Group #2 | Group #3 | Total |
|---|---|---|---|---|
| NN-SE | 2.68 | 2.39 | 2.59 | 7.65 |
| THREE-LAYER NN-SE | 2.78 | 2.35 | 2.46 | 7.59 |
| PROPOSED MODEL | 2.92 | 2.06 | 2.34 | **7.33** |

表 5.4: 各グループの誤りの平均個数.

| Model | Group #1 | Group #2 | Group #3 |
|---|---|---|---|
| NN-SE | 34.9 | 29.9 | 35.3 |
| THREE-LAYER NN-SE | 36.7 | 29.7 | 33.5 |
| PROPOSED MODEL | **40.3** | 26.7 | 33.1 |

表 5.5: 各グループの誤りの平均割合 (%).

**GOLD:**
10, 13, 19, 25, 76, 118, 120, 122, 164

---

**THREE-LAYER NN-SE:**
23, 25, 26, 38, 47, 106, 108, 110, 118, 162, 163, 164, 169, 170

---

**PROPOSED MODEL:**
25, 26, 72, 79, 106, 118, 119, 120, 121, 161, 162, 163, 164

図 5.6: GOLD に対して, THREE-LAYER NNSE および PROPOSED MODEL が選んだ文集合の比較. 青色の数字はモデルが正解した文, 下線付きの緑色の数字は不正解だったが Group #1 に属している文, 波線付き赤色は Group #3 に属している文を表す.

一方で, 逆に 31%程度しか回収出来ていないにも関わらず, ROUGE スコアで見ると 50%程度のユニグラムの被覆度があることがあることに着目したい. これは, アブストラクトが持つ単語を含む負例の文が多く存在していることを表している. 論文においては同じような意味の文が複数回登場したり, 文中で重要となるキーワードが多くの文で登場するといった特性があり, そのような性質が原因として考えられる.

次に, システムが誤って選んだ文を分類した時の各グループの平均個数と割合をそれぞれ表 5.4, 5.5 に示す. これを見ると, 文のみに着目している NN-SE および THREE-LAYER NN-SE と比べて, PROPOSED MODEL はトータルの誤りの個数が少なくなっただけでなく, 内訳としても Group #1 に属する文が増加したことが分かり, 提案手法が効果的に作用したことが分かる.

## 5.7 考察

提案手法によって期待される効果の一つは, 関連性の薄い文の抽出を防ぐことであった. 図 5.6 に階層的なスコア計算が効果的に作用した例を示す. この例では正解数はそれぞれ 3 つと 4 つで一つしか違わないが, ROUGE-2 の F 値を見ると前者が 16.6 なのに対し, 後者は 25.7 と大きく異なっている. この差として Group #1 に属している文が THREE-LAYER NNSE は 3 つしかないのに対し PROPOSED MODEL は 8 つ含まれていることが原因であると考えられる. 反対に, THREE-LAYER

**Abstract:**
Measuring teamwork requires identifying dimensions of teamwork or processes that comprise the teamwork construct, while taskwork requires identifying specific team functions.

---

**Sentence #76 (GOLD):**
We further categorized the selected factors into dimensions of teamwork , or processes that comprise the teamwork construct .

---

**Sentence #72 (PROPOSED MODEL):**
We used a weighting system to select factors for measuring teamwork from those identified and sorted by the participants .

---

**Sentence #79 (PROPOSED MODEL):**
We categorized these factors -LRB- determinants -RRB- into three groups : personal , community-related and service-related .

---

**Sentence #38 (THREE-LAYER NN-SE):**
A total of 36 individuals were involved .

---

**Sentence #47 (THREE-LAYER NN-SE):**
The timeline activity initiated dialogue on teamwork .

図 5.7: 教師信号の文#76 に対して, 提案モデルが選んだ文#72 と#79 および THREE-LAYER NN-SE が選んだ文#38 と#47 の比較.

NNSE が間違って選んだ文の多くが Group #3 に属している文であり, それだけ無関係な文を多く選んでしまっている. 具体的に中身を見ると, 提案モデルが選んだセンテンス#72 およびセンテンス#79 は正解文#76 と同一のパラグラフにあるが, THREE-LAYER NNSE ではこれと同一の段落からは文を選べていない.

図 5.7 に正解文#76 とそれと最も近いことを述べていると考えられるアブストラクト中の一文, そして提案モデルが選んだセンテンス#72 およびセンテンス#79, それに対して既存モデルが選んだセンテンス#38 およびセンテンス#47 の具体的な内容を示す. この論文の大まかな趣旨は, 発展途上国の医療現場において, 医療に従事する人たちのチームがどれほど機能するかを様々な指標で測定するというものである. アブストラクト中の一文は, チームワークを評価するにはチームの関係性が構成されていく過程を見なければならず, タスクワークを評価するにはそのチーム特定の機能を見なければならないと述べいている. それに対し, 正解文である文#76 では, 考察している要因のうちいくつかを「チームワークの関係性の構築段階に寄与している」というカテゴリーの中に入れたと述べており, アブストラクト中の一文の前半部分と関係している文だとわかる. 一方で, 文#72 では参加者

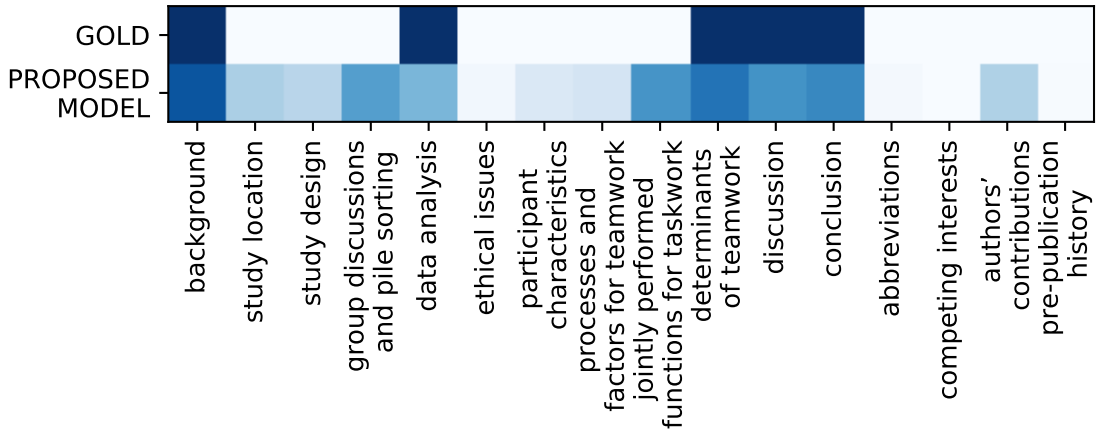


図 5.8: セクションの正解ラベルと PROPOSED MODEL が出力した各セクションのスコアの比較.

**GOLD:**
0, 4, 19, 22, 64, 82, 148, 153, 192, 193

---

**PROPOSED MODEL:**
0, 11, 14, 22, 44, 148, 150, 169, 184, 233, 234, 235, 236

図 5.9: GOLD と PROPOSED MODEL が選んだ文集合の比較例. 青色の数字が一致した文, 二重下線付きの紫色の数字が間違えている上にそれが連続してしまった文を表す.

から得られた様々な要因に対して重みづけを行ったと述べていて, 文#79 ではこれらの要因を個人的なものか, 地域社会に関係したものか, サービスに関係したものかに分類したと述べている. システムが選んだ 2 つの文は「チームの関係性の構築」というキーワードこそ含まれていないが,「チームワークを測定するための要因を解析した」というトピックについては拾えていることが分かる. それに対して, THREE-LAYER NN-SE が間違えて選んだ文を見ると, 36 人の被験者が実験に参加したことや, 実験中のグループディスカッションにおける対話の様子についてなど, 実験の細かい内容について選んでしまっており, ファクターの分類の話とはピントがずれていることが分かる.

次に, このサンプルにおいて PROPOSED MODEL が計算した各セクションのスコアを可視化したものを図 5.8 に示す. ここでは色が濃いほどスコアが 1 に近かったことを表す. この図を見ると, セクションの正解ラベルに対して大まかには正しく予測できていることが分かる. 例えば先頭の“background”や, 後半の“determinants of teamwork”,“discussion”,“conclusion”の三連続が重要であるという判断をモデルが下せていることが分かる. また, 先ほどの例で THREE-LAYER NN-SE

が選んだ文#38 は“ study design ”のセクションに属する文だが, 相対的にこのセクションは低くスコアリングされていることが分かる. 一方で“ author's contribution ”のセクションはなぜかそれなりに重要だと判断してしまっている. 結果的にこのセクションからは文は抽出していないものの, セクションの重要度の予測が必ずしも成功しているわけではない.

最後に, 特徴的な間違いのパターンとして, 図 5.9 に挙げた例のように連続して文を間違えてしまうというものが見受けれられた. 考えられる原因としては, そもそも教師信号として設けた文集合のうち, このように連続した文を含んでいるものがあったため, その連続的な文の選択という挙動をモデルが学習してしまった可能性が考えられる. すなわち, 連続して文に高いスコアを付与すること自体は必ずしも間違いではないものの, それが見当はずれのポイントからスタートしてしまったために, 図 5.9 に記載されているようなパターンが起きてしまったのだと考えられる.

# 第6章　結論

## 6.1　本論文のまとめ

本論文では, 教師有り学習に基づく, 学術論文を対象とした抽出型の要約モデルを構築した. 学術論文のサイズが大きいことに起因して, リカレントニューラルネットワークの処理精度が劣化することと抽出文候補が膨大であることがこのタスクの問題点であった. 前者に対してはリカレントニューラルネットワークを階層的に設計したモデルを提案した. 後者に対しては文書構造を木構造とみなして, 文だけでなくパラグラフとセクションにも正解ラベルを付与し, かつ各ノードのスコア計算の際に親ノードの情報を参照することによって, より正例ノード中の木の中にモデルが向くように誘導する手法を提案した. また予備実験によって正例文を含んでいる段落の方がアブストラクトと関連性が高く, 反対に正例文を全く含んでいないセクション中の段落はアブストラクトとの関連性が低い傾向にあることを定量的に示し, 二つ目の提案手法によってモデルが負例文を選んでしまった場合でも, より正例文に近い文脈から選ぶように期待した.

生命医学系の論文を要約データセットとした実験を行い, 提案モデルが既存モデルよりも高い精度を達成したことを示した. また, モデルが正解文をより多く拾えていること, つまりモデルが学習データから文の選択パターンをより習得できていることを確認した. さらに, 文の間違いの中身を見ても, 提案モデルの方がより正例文と近い文脈から抽出している傾向があることが分かった.

## 6.2　今後の課題

本論文では学術論文を対象としてモデルを設計したが, 提案モデルは論文以外の文書構造をもつテキストにも適用可能であると考えられる. 例えば Wikipedia など, ある程度サイズが大きくかつ章構造を有したデータセットについても提案モデルを構築しその有効性を確認したい.

また, 図 4.1 に示したように学術論文は単なる階層構造ではなく多分木の木構造としてとらえられるが, 多分木や二分木に特化した Tree LSTM を Tai ら [29] が提案しており, そのような LSTM を用いることでより木構造に沿ったモデルが構築できる. 今回の実験設定ではセクション内にサブセクションが含まれているような構造は無視し, それぞれ独立したセクションとみなしたので, そういった包含関係も考慮したモデルを組むうえでも Tree LSTM は有効であると考えられる.

また, 今回の実験では実験データを 30,000 本としたが, ニューラルネットワークモデルをよりパワフルに活用するため, より大きな学習データを適用し精度の改善を試みたい.

別の観点としては計算量の削減に取り組みたい. 例えば関連研究として, 長い文書を RNN に読み込ませるときに不必要な文は読み飛ばすことを学習させるものがある [36]. 予備実験によって論文に

は重要な箇所と重要でない箇所があることは先に述べた. 提案モデル中でセクション, パラグラフのスコアを計算することはどれくらいその文脈が重要かを予測することに相当するが, そのスコアを利用してあらかじめ重要でない箇所はそもそも LSTM に入力しないことで, 計算量を大幅に削減できると考えられる.

# 参考文献


[1] 奥村学, 難波英嗣. テキスト自動要約. オーム社, 2005.

[2] Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pp. 1106–1115, 2015.

[3] Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1073–1083, 2017.

[4] Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. Faithful to the original: Fact-aware neural abstractive summarization. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, page to appear, 2018.

[5] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. Summarunner: An interpretable recurrent neural network model for extractive summarization. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pp. 3075–3081, 2017.

[6] Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 17–21, 2015.

[7] Preksha Nema, Mitesh M. Khapra, Anirban Laha, and Balaraman Ravindran. Diversity driven attention model for query-based abstractive summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1063–1072, 2017.

[8] Mikael Kågebäck, Olof Mogren, Nina Tahmasebi, and Devdatt Dubhashi. Extractive summarization using continuous vector space models. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality @ European Chapter of the Association for Computational Linguistics*, pp. 31–39, 2014.

[9] Jianpeng Cheng and Mirella Lapata. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pp. 484–494, 2016.

[10] Masaru Isonuma, Toru Fujino, Junichiro Mori, Yutaka Matsuo, and Ichiro Sakata. Extractive summarization using multi-task learning with document classification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2101–2110, 2017.

[11] Shashi Narayan, Nikos Papasarantopoulos, Mirella Lapata, and Shay B. Cohen. Neural extractive summarization with side information. *arXiv preprint arXiv:1704.04530*, 2017.

[12] Baotian Hu, Qingcai Chen, and Fangze Zhu. Lcsts: A large scale chinese short text summarization dataset. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1967–1972, 2015.

[13] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan:sequence generative adversarial nets with policy gradient. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pp. 2852–2858, 2017.

[14] Danish Contractor, Yufan Guo, and Anna Korhonen. Using argumentative zones for extractive summarization of scientific articles. In *Proceedings of the 23th International Conference on Computational Linguistics*, pp. 663–678, 2012.

[15] Arman Cohan and Nazli Goharian. Scientific article summarization using citation-context and article's discourse structure. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 390–400, 2015.

[16] Ed Collins, Isabelle Augenstein, and Sebastian Riedel. A supervised approach to extractive summarisation of scientific papers. In *Proceedings of the 21st Conference on Computational Natural Language Learning*, pp. 195–205, 2017.

[17] Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*, 2015.

[18] Yizhe Zhang, Dinghan Shen, Guoyin Wang, Zhe Gan, Ricardo Henao, and Lawrence Carin. Deconvolutional paragraph representation learning. In *Advances in Neural Information Processing Systems 30*, pp. 4172–4182, 2017.

[19] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1480–1489, 2016.

[20] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, 2015.

[21] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Proceedings of the ACL 2004 Workshop on Text Summarization Branches Out*, pp. 74–81, 2004.

[22] Marc'AurelioRanzato, Sumit Chopra, Michael Auli, Wojciech Zaremba. Sequence level training with recurrent neural networks. In *Proceedings of the 4th International Conference on Learning Representations*, 2016.

[23] Ayana, Shiqi Shen, Zhiyuan Liu, and Maosong Sun. Neural headline generation with minimum risk training. *arXiv preprint arXiv:1604.01904*, 2016.

[24] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017.

[25] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pp. 3111–3119, 2013.

[26] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, Vol. 9, No. 8, pp. 1735–1780, 1997.

[27] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, pp. 3104–3112, 2014.

[28] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1412–1421, 2015.

[29] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1556–1566, 2015.

[30] Izzeddin Gur, Daniel Hewlett, Alexandre Lacoste, and Llion Jones. Accurate supervised and semi-supervised machine reading for long documents. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2011–2020, 2017.

[31] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning*, Vol. 37, pp. 2048–2057, 2015.

[32] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1746–1751, 2014.

[33] 中須賀謙吾, 鶴岡慶雅. 談話構造を利用した学術論文の自動要約生成. 言語処理学会第 21 回年次大会 発表論文集, pp. 569–572, 2015.

[34] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, pp. 2342–2350, 2015.

[35] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, Vol. 15, No. 1, pp. 1929–1958, 2014.

[36] Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Illia Polosukhin, Alexandre Lacoste, and Jonathan Berant. Coarse-to-fine question answering for long documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 209–220, 2017.

# 発表文献

## 査読付き会議論文

1. Kazutaka Kinugawa and Yoshimasa Tsuruoka. Developing a Supervised Text Summarizer with Academic Papers in Biomedical Sciences. *In Proceedings of First International Workshop on ScIentific Document Analysis,* 2016.

2. Kazutaka Kinugawa and Yoshimasa Tsuruoka. A Neural Hierarchical Extractive Summarizer for Academic Papers. *In Proceedings of Second International Workshop on Scientific Document Analysis,* 2017.

## 査読無し会議論文

1. 衣川和尭, 鶴岡慶雅. 学術論文の章構造に基づくニューラル自動要約モデル. 言語処理学会第 23 回年次大会発表論文集, pp.150–153, 筑波, 2017 年 3 月.

# 謝辞

本研究を進めるにあたり多くの方々にお世話になりました.

指導教員である鶴岡慶雅准教授には, 修士課程の 2 年間にわたりお世話になりました. 私は全く異なる分野から今の研究室に配属になったため, 最初は分からないことだらけでしたが, 言語処理初学者が読んだ方が良い教科書を教えていただくところから始まり, 最初のチュートリアルでは非常に基礎的なことを教えていただきました. 研究が始まってからは, 研究の方針や具体的な内容についてのアドバイスはもちろん, 英語の勉強法や英語論文の修正など何から何まで丁寧に指導していただきました. 特に私は締め切りギリギリまで原稿が終わらないことが多くご迷惑をお掛けしましたが, いつも時間を割いて修正をくださりました. 数々のご指導ありがとうございました, 感謝を申し上げます.

研究室の先輩である江里口瑛子さん, 亀甲博貴さん, 橋本和真さん, 水上直紀さんには, 研究に関する事細かなアドバイスはもちろんのこと, この分野の最先端のお話や海外でのインターンのお話をしていただいて非常に励みになりました. また、研究や計算機のことなど分からないことを質問するといつも快く答えてくださり, 非常に頼もしい先輩方でした. 二年間ありがとうございました.

研究室の後輩の皆さんも非常に優秀な方が多く, 私の研究に関して様々なコメントをくれたり, 学術的なディスカッションを色々交わして勉強になることが多かったです. いつもありがとうございました.

同期の田口直弥君, 扇本岳大君, シュイ ヤンさんとは互いに切磋琢磨しながら充実した研究生活を共に送りました. いろいろご迷惑をおかけすることもあったかもしれませんが, とても楽しい 2 年間でした. 本当にありがとうございました.

最後に今まで長いこと学生であることを認め, 支えてくれた家族に感謝いたします. ありがとうございます.

平成 30 年 2 月 1 日