音声反訳を用いた日本語文章の難易度推定

相澤美悠　指導教員:西村俊二

令和2年1月27日

Difficulty Estimation of Japanese Sentence Using Speech Transcription

MIYU AIZAWA　(ACADEMIC ADVISOR　SHUNJI NISHIMURA)

概要：文章は，日々の生活の中で幅広く用いられている情報伝達手段である．人々は，情報がよりわかりやすい文章で表されることを求めている．しかし，文章の難しさの基準は人によって異なるため，わかりやすい文章を書くことは容易ではない．そこで，文章作成支援を目的として，文章の難易度を推定する研究が行われてきた．現在，日本語文章のリーダビリティを測定するシステムは存在するが，まだまだシステムによる可読性評価と人間の可読性評価にはギャップがある．つまり，システムによる可読性評価で読みやすいとされた文章が人間にとっては難しい場合があるため，日本語のリーダビリティを測定するシステムは十分とは言えない．そこで本研究では日本語文章の読みやすさ（音読のしやすさ），聞き取りやすさを指標にして日本語文章の可読性を推定する手法を提案する．さらに，日本語文章の難易度推定に音声反訳を用いることが有効であるか小学1年生，小学4年生，小学5年生，中学3年生で習う各難易度の国語の教科書に載っている作品の文章に対しての評価実験を行い考察した．

キーワード: 音声反訳，難易度，リーダビリティ

1. 緒言

1.1 研究背景

文章の難易度に関する研究は，リーダビリティ研究と深く関わっている．リーダビリティとは文章の可読性のことであり，英語におけるリーダビリティ研究では，Flesch Reading Ease（FRE）というリーダビリティ指標が標準的指標として確立しているなど古くから先行研究[1] があり，Microsoft Word の文章校正機能に利用されていたりする．日本語におけるリーダビリティ研究では，標準的指標は未確立だが，システムの開発事例として「帯」[2] や「jReadability」[3] が存在する．リーダビリティ研究は，文章の特徴を数量的に評価しようとするものである．つまり，文章の書き手による文体を客観的に評価することで読み手の評価を捉えるものである．

リーダビリティ研究の主な手法としては，各難易度で分類された語彙リストを作成し，文章中の語彙の割合を用いる方法がある. また, 文の長さや品詞の分布，句読点の頻度などの文章の表層的な情報をもとにリーダリティ公式という計算式によりリーダビリティを求める方法がある．さらに，文字の連続をもとに言語処理を行う言語モデル bigram などを用いる方法がある．「帯」は言語モデルによる手法で日本語文章の難易度推定を行っており[4] ，「jReadability」は以下のリーダビリティ公式により日本語文章の難易度推定を行っている[5].

$$X = \{\text{平均文長} * -0.056\} + \{\text{漢語率} * -0.126\}$$
$$+ \{\text{和語率} * -0.042\} + \{\text{動詞率} * -0.145\}$$
$$+ \{\text{助詞率} * -0.044\} + 11.724$$

しかし，システムと被験者の文章難易度評価の比較実験の結果[6] に示されているように，難語数が多いにも関わらず，芥川龍之介の“羅生門”のようにシステムによる評価が読みやすいとされているが，人による評価は読みにくいとされている文章もある．

1.2 目的

本研究では，音声反訳を用いることで日本語文章の読みやすさ（音読のしやすさ），聞き取りやすさを指標にして日本語文章の難易度を推定する手法の提案を行い，日本語文章の可読性評価に音声反訳を用いることが有効であるかどうかの評価実験を行う．日本語文章の読みやすさ（音読のしやすさ），聞き取りやすさを指標にする理由は，音声データが文章になる過程で音声データの音素を特定し，音素を辞書でマッチングし，単語として表現する．さらに単語のつながり状況を調べ，最も可能性の高い組み合わせで文章にする．このことから，読みやすい文章ほど正しい音声データになり，よく使われている表現ほど正しい文章になると考えられるからである．

1.3 本論文の構成

本論文の構成は以下の通りである．2 章では提案手法で用いる API やソフトウェア，ライブラリなどの概要，または実際にその技術を用いた際の動作について述べる．3 章では，提案手法の具体的な内容について述べる．4 章では，3 章で提案した手法の精度を調べるための評価実験の手法について述べる．また実験結果として，文章の難易度と提案手法の比較を行い，その結果について考察を行う．5 章では本研究の今後の課題について述べる．6 章では本論文の結論をまとめる．

2. 本研究で用いた技術

本章では，WebSpeechAPI，jsdiff，kuromoji.js，正規表現について述べる．2.1 では WebSpeechAPI の概要と，文章に対してどのように音声反訳を行うかについて説明する．2.2 では，diff の概要と動作について説明する．2.3 では，kuromoji の概要と動作について説明する．2.4 では，正規表現を用いてひらがなやカタカナ，漢字を区別する方法について説明する．

2.1 Web Speech API

本研究では音声反訳に Web Speech API［7］を用いた．Web Speech API とは，W3C（World Wide Web Consortium）によって定義されている，音声合成と音声認識を実現する Web アプリケーションを開発することができる Javascript API である．具体的には，音声合成用の Web Speech Synthesis API と，音声認識用の Web Speech Recognition API の 2 つで Web Speech API と定義されている．音声合成と音声認識の内容については次の 2.1.1 と 2.1.2 で述べる．

2.1.1 音声合成

Web Speech API の音声合成は，SpeechSynthesis インターフェイス経由でアクセスされる．これは，プログラムにそのテキストコンテンツを読み上げる機能を提供しているものである．また，読み上げる音声の種類については，SpeechSynthesisVoice オブジェクトで表され，ブラウザによって変更可能である．現在，PC では Google Chrome, Microsoft Edge, Mozilla Firefox, Opera, Safari に対応している［8］．

2.1.2 音声認識

Web Speech API の音声認識は，SpeechRecognition インターフェイス経由でアクセスされる．これは，音声入力から音声の文脈を認識し，適切に応答する機能を提供しているものである．

現在，PC では Google Chrome 対応している［8］．そのため，本研究では Google Chrome で動かすことを前提とした手法を提案する．

実際に音声認識を行った結果が表 1 のとおりである［9］，［10］．

表 1．音声認識した結果

文章 A	生麦生米生煙草
音声認識した結果	生むぎ 生米を タバコ

2.2 jsdiff

日本語の文章を表すテキストと音声認識した結果のテキストの差分を求めるために diff を用いる．diff とは 2 つの文章に対して差分を求めることである．具体的には，2 つの文章に対して，変化がないのか，文字列が追加，削除されたのかを編集距離や LCS，SES を計算することで差分を求める．この編集距離とは，単語を 1 つの文字とみなし，相異なる 2 つの単語がそれぞれ別の文字として追加や削除の処理をして求められるもののことである．また，LCS は 2 つの文章の最長の共通部分のことであり，SES は文章 X から文章 Y へ変換するための最短手順のことである．

本研究では jsdiff［11］を用いる．jsdiff とは文字，単語，行単位の差分をチェックする JavaScript 製のオープンソースソフトウェアである．jsdiff は，より高速に編集距離や SES を求めるアルゴリズムとして，Myers による O(ND) アルゴリズム［12］を元に実装されているものである．N は要素 X，Y の合計の長さ，D は編集距離を表している．

jsdiff の関数 **diffChars** を用いた例が以下のとおりである．

Diff.**diffChars**(元の文章, 新しい文章)

関数 **diffChars** により，2 つのテキストブロックを比較し，文字ごとの比較を行い，以下の形式で返す．

{
oldStr ：元の文字列の diff
newStr ：新しい文字列の diff
}

実際に 2 つの文章に対して diff を行った結果を次の 2.2.1，2.2.2 で述べる．

2.2.1 例 1

下記の 2 つの文章 B，C に対して diff を行った結果が図 1 である．

・文章 B：私は親です
・文章 C：私が親でした

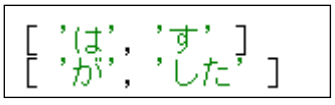


図 1．diff を行った結果

しかし，日本語の文章の中にはひらがなや漢字が組み合わさっているため，diff されたとき意味が同じ場合がある．具体例として，次の 2.2.2 で述べる．

2.2.2 例 2

2.2.1 の文章 B，下記の文章 D に対して diff を行った結果が図 2 である．

・文章 D：わたしはおやです

```
[ '私', '親' ]
[ 'わたし', 'おや' ]
```

図 2．diff を行った結果

文章 B と文章 D は同じ意味の文章である．しかし，文章 B は漢字を用いて表記されているが，文章 D はすべてがひらがなで表記されている．このように表記が異なると文章の意味は同じでも diff されてしまう．この問題に対しての解決方法を次の 2.3，2.4 で述べる．

2.3 kuromoji.js

diff したものに対して，ひらがなや漢字など意味は同じでも表記が異なる場合，形態素解析を行いその形態素の読みを比べることでこの問題が解決できると考えた．形態素解析には kuromoji.js[13] を用いた．kuromoji.js とは，Java の形態素解析器である Kuromoji を JavaScript で使えるようにしたものである．実際に文章 B に対しての解析結果[14] を表 2 に示す．

表 2．“私は親です”の形態素解析結果

品詞，品詞細分類 1，品詞細分類 2，品詞細分類 3，活用型，活用形，基本形，読み，発音	
私	名詞，代名詞，一般，*，*，*，私，ワタシ，ワタシ
は	助詞，係助詞，*，*，*，*，は，ハ，ワ
親	名詞，一般，*，*，*，*，親，オヤ，オヤ
です	助動詞，*，*，*，特殊・デス，基本形，です，デス，デス

本研究では，形態素解析結果の基本形と読みを用いる．

下記の 2 つの文章 E，F に対して diff を行った結果が図 3 である．

・文章 E：わたしはおやです

・文章 F：私が親です

```
[ 'わたしはおや' ]
[ '私が親' ]
```

図 3．diff を行った結果

図 3 の diff を行った結果に対して形態素解析による形態素と読みが図 4，図 5 である．図 5 の結果でもう一度 diff を行った結果が図 6 のとおりである．

```
[ 'わたし', 'は', 'おや' ]
[ '私', 'が', '親' ]
```

図 4．形態素解析結果（形態素）

```
[ 'ワタシ', 'ハ', 'オヤ' ]
[ 'ワタシ', 'ガ', 'オヤ' ]
```

図 5．形態素解析結果（読み）

```
[ 'ハ' ]
[ 'ガ' ]
```

図 6．diff を行った結果

この手法の問題点は，形態素解析の読みが実際の文章中での読みの通りにならない場合があることである．例えば，“植”という漢字は文章中では“う”と読む場合でも，形態素解析の読みでは“うえ”と読む場合がある．このことについては，5 章の今後の課題で述べる．

実際に，ひらがなとカタカナと漢字をどのように区別しているかは次の 2.4 で述べる．

2.4 正規表現

diff したものに対して，ひらがなやカタカナ，漢字の表記が異なるかどうかの判定は正規表現[15] 用いた．具体的には，文字列内で文字の組み合わせを照合するために用いられるパターンが条件に当てはまるかを判定した．まず，ひらがな，カタカナ，漢字を表す UTF8 の範囲[16] について表 3 に示す．

表 3．UTF8 の範囲

ひらがな	0x3041～0x3093
カタカナ	0x30A1～0x30F6
漢字	漢字の踊り字（々）：3005 漢数字のゼロ：3007 漢字の踊り字（〻）：303b CJK 統合漢字（拡張 A）：3400～9FFF CJK 互換漢字：F900..FAFF

表 3 を踏まえた上で，次にひらがなやカタカナ，漢字を表す正規表現を表 4 に示す．

表 4．正規表現

ひらがな	/[¥u{3000}-¥u{301C}¥u{3041}-¥u{3093}¥u{309B}-¥u{309E}]/mu
カタカナ	/[¥u{3000}-¥u{301C}¥u{30A1}-¥u{30F6}¥u{30FB}-¥u{30FE}]/mu
漢字	/([¥u{3005}¥u{3007}¥u{303b}¥u{3400}-¥u{9FFF}¥u{F900}-¥u{FAFF}¥u{20000}-¥u{2FFFF}][¥u{E0100}-¥u{E01EF}¥u{FE00}-¥u{FE02}]?)/mu

2.3，2.4 の手法を用いることで，diff された文章の表記が異なる場合は形態素解析により，その形態素の読みで比べる手法をとることにする．この手法の問題点

は，漢字とひらがなの読みが同じでも文章的には意味が異なる場合があることである．例えば，“かぶ”と“株”は“カブ”と読むが，食べ物のカブの場合“蕪”という漢字を使う．つまり読みで比べることで，漢字の意味は異なるなどの問題もある．このことについては，5 章の今後の課題で述べる．

3. 提案手法

本章では，本研究で提案する日本語文章の難易度推定手法の概要，各難易度の文章における差異率による日本語文章の難易度推定方法について述べる．3.1 では日本語文章の難易度推定を行うまでの流れと実際にブラウザで実行したことについて説明する．3.2 では，本研究での難易度の定義と難易度推定を行うための差異率の取得について説明する．

3.1 提案手法の概要

3.1.1 日本語文章の難易度推定を行うまでの流れ

本研究で提案する日本語文章の難易度推定手法の概要について説明する．入力データは，難易度を評価したい日本語の文章を表すテキストであり，出力データは，推定した難易度である．また，日本語文章の難易度推定を行うまでの流れを図 7 に示す．

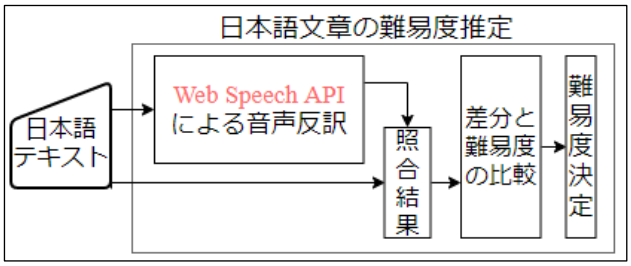


図 7．難易度推定までの流れ

図 7 は，まず日本語の文章を表すテキストを入力する．その後，Web Speech API により入力したテキストの音声反訳によるテキスト化を行う．これら 2 つのテキストを jsdiff によって照合し差分を抽出する．この差分を用いて文章の難易度推定を行う．具体的な難易度推定の方法を次の 3.2 で述べる．

実際にブラウザ上でどのように実行されているかは次の 3.1.2 で述べる．

3.1.2 ブラウザでの実行

実際にブラウザ上で実行したものを図 8 に示す．図 8 の概要を以下に述べる．

図 8 のテキストボックスに難易度を評価したい日本語の文章を表すテキストを入力し，start ボタンを押すと Web Speech API による音声合成が始まる．その後の実行結果を表 5 に示す．

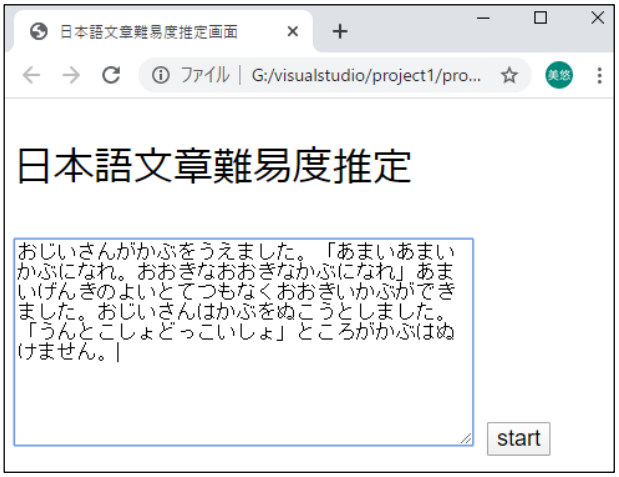


図 8．ブラウザ上での実行

表 5．ブラウザ上での実行結果

テキスト	diff
1 おじいさんがかぶをうえましたあまいあまいかぶになれおおきなおおきなかぶになれあまいげんきのよいとてつもなくおおきいかぶができましたおじいさんはかぶをぬこうとしましたうんとこしょどっこいしょところがかぶはぬけません	かぶ，う，あま，あま，げんき，よ，おお，かぶ，かぶ，ぬ
2 おじいさんが株を植えました甘い甘いかぶになれおおきなおおきなかぶになれあまい元気の良いとてつもなく大きいカブができましたおじいさんはかぶをぬこうとしましたうんとこしょどっこいしょところがカブは抜けません	株，植，甘，甘，元気，良，大，カブ，カブ，抜
読み	diff
1 カブ，ウ，アマ，アマ，ゲンキ，ヨ，オオ，カブ，カブ，ヌ	オオ
2 カブ，ウエ，アマ，アマ，ゲンキ，ヨ，ダイ，カブ，カブ，ヌ	エ，ダイ
106 文字	

表 5 の 2 行目は照合する 2 つのテキストとそれらを diff した結果を表示している．テキスト 1 は，入力テキストから句読点などを削除したテキストを表示している．これは，次の 3.1.3 で述べる．テキスト 2 では，音声反訳した結果のテキストを表示している．そして，表 5 の 4 行目では 2 行目の diff した結果のひらがなやカタカナ，漢字など表記の違うものの読みとそれらを diff した結果を表示している．しかし，ここで読み 2 の

読みが間違っている場合や，読みで diff を行ったために文章中では正しくない漢字が diff されている場合があるためその判定は手動で行っている．最後に表 5 の 5 行目でテキスト 1 の文字数を表示している．これは次の 3.2.2 で述べる差異率を求めるのに用いる．

3.1.3 テキストの整理

入力テキストから句読点などの削除は replace()メソッド[17] を用いている．そのプログラムをソースコード 1 に示す．

ソースコード 1　replace

```
var retext1 =　text1.replace(/、/g, ‘’)
                .replace(/。/g, ‘’)
                .replace(/「/g, ‘’)
                .replace(/」/g, ‘’)
```

ソースコード 1 の replace により//で囲まれた句読点や鉤括弧を空白文字に置換している．

このテキストの整理を行う理由としては，音声認識したテキストには句読点などが含まれないため，入力テキストのままだと句読点などが diff されてしまうからである．

3.2 難易度推定

3.2.1 難易度の定義

本研究では，日本語の文章に対しての難易度推定することを目的としているが，そもそも文章自体に固有の難易度は存在しない．その文章の可読性を難易度という仮定のレベルで置き換えているだけである．つまり文章の難易度とは，平易や難しいなどの軸において文章を分類するための仮想的なスケールのことである．

本研究では，小中学校で使われる国語の教科書がすでに難易度の付与されたものと考えて，日本語の文章に対して学年という難易度を付与する．例えば，小学校 1 年生向けの国話の教科書に載っている作品の文章は小学 1 年レベルの難易度であると考える．

3.2.2 差異率とは

本研究では，文章の難易度推定に差異率を用いる．この差異率とは，入力された文章と音声反訳を行った後の文章の差分を diff で求めるが，この差分の率のことである．差異率は以下の式で求める．

差異率(%)＝差分の文字数/入力文章の文字数

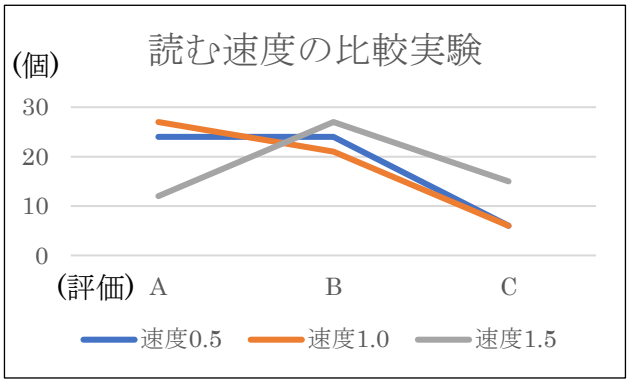
3.2.3 音声反訳を行う上での精度について

Web Speech API を用いて音声合成と音声認識を行う上で，入力テキストに対する読む速度についての比較実験を行った．具体的には，小学 1 年の教科書に載っている作品「大きなカブ」，小学 4 年の教科書に載っている作品「ごんぎつね」，中学 3 年の教科書に載っている作品「羅生門」に対して，読む速度を 0.5，標準速度 1.0，速度 1.5 の 3 つの速度で比較実験を行った．さらに，各速度に対して 3 回ずつ差異率を測定した．

測定した差異率について次の A，B，C で評価を行った．

A：3 回のうち 3 回とも同じ値
B：3 回のうち 2 回が同じ値
C：3 回のうちすべてが違う値

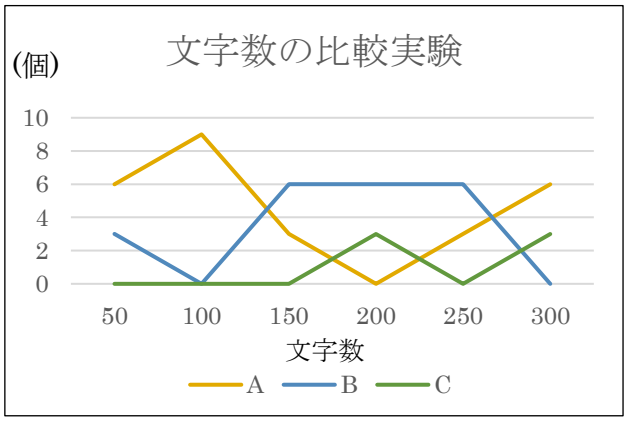
その結果をグラフ 1 に示す．



グラフ 1．読む速度比較実験の結果

グラフ 1 の結果から，評価の高い A が一番多く，評価の低い C が一番少ない標準速度の 1 を読む速度とする．

次に，文章の文字数に対しての比較実験を行った．この実験を行う理由としては，本研究では音声合成の後に音声認識を行っているため文字数が増えると正しく音声反訳できなくなることがある．そこで，文字数を制限することで，その問題による影響を最小限に抑えるためである．文字数比較実験の結果をグラフ 2 に示す．



グラフ 2．文字数比較実験の結果

グラフ 2 より，150 文字以上になると評価 A の数が少なくなり，評価 C の数が多くなりはじめるので本研究では，150 文字で 1 つの文章を制限することにする．

以上のことより文章を 150 文字以内に分割した上で文章の差異率の測定を行う．

3.2.4 差異率による難易度推定

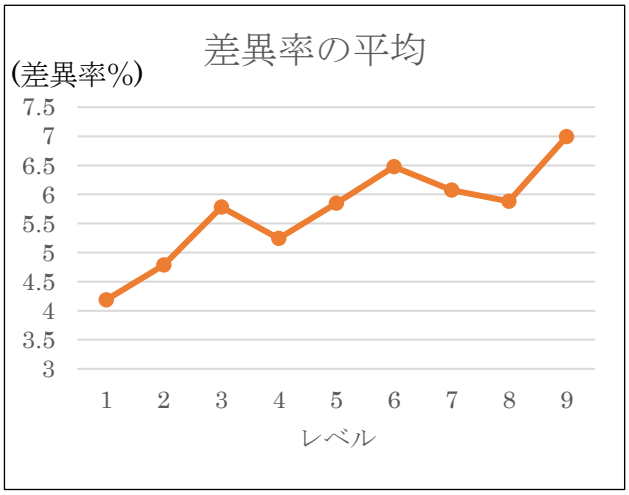
差異率の測定を行う際に，まず小学 1 年から小学 6 年の教科書に載っている作品と中学 1 年から中学 3 年に載っている作品を集めた．具体的には，160 作品，762 個の文章を集めた．

各レベルの具体的な文章例を表 6 に示す．

表 6. 各レベルの文章例

学年	レベル	文章例
小学 1 年	1	おじいさんがかぶをうえました。「あまいあまいかぶになれ。おおきなおおきなかぶになれ」あまいげんきのよいとてつもなくおおきいかぶができました。
小学 2 年	2	広い海のどこかに、小さな魚のきょうだいたちが、たのしくくらしていた。みんな赤いのに、一ぴきだけは、からす貝よりもまっくろ。およぐのは、だれよりもはやかった。
小学 3 年	3	私が両手をひろげても、お空はちっとも飛べないが飛べる小鳥は私のように、地面を速くは走れない。
小学 4 年	4	これは、わたしが小さいときに、村の茂兵というおじいさんからきいたお話です。むかしは、わたしたちの村のちかくの、中山というところに小さなお城があって、中山さまというおとのさまがおられたそうです。
小学 5 年	5	小さな蜘蛛が一匹、路ばたを這って行くのが見えました。そこで陀多は早速足を挙げて、踏み殺そうと致しましたが、「いや、いや、これも小さいながら、命のあるものに違いない。」
小学 6 年	6	原爆ドームが世界遺産の候補として、世界の国々の審査を受けることになったとき、わたしは、ちょっぴり不安を覚えた。
中学 1 年	7	銀木犀の花は甘い香りで、白く小さな星の形をしている。そして雪が降るように音もなく落ちてくる。
中学 2 年	8	メロスは激怒した。必ず、かの邪智暴虐の王を除かなければならぬと決意した。メロスには政治がわからぬ。メロスは、村の牧人である。
中学 3 年	9	高瀬舟は京都の高瀬川を上下する小舟である。徳川時代に京都の罪人が遠島を申し渡されると、本人の親類が牢屋敷へ呼び出されて、そこで暇乞いをすることを許された。

各レベルの差異率の平均をグラフ 3 に示す．



グラフ 3. 差異率の平均

グラフ 3 より，各レベルの差異率の平均にはあまり差がでなかった．

次に，各レベルの差異率に対して，難易度を推定することの有効性を調べるために，各レベルと差異率の相関係数を求めた．その結果が表 7 のとおりである．

表 7. 各レベルと差異率の相関係数

レベル別	相関係数
レベル 1〜9	0.186 （ほとんど相関がない）
レベル 1，9	0.533（相関がある）
レベル 1，4，5	0.358（少し相関がある）
レベル 1，4，5，9	0.431（相関がある）

4 つのレベルにおいては，表 7 の 5 行目の 4 つのレベルのとき最も相関係数が大きいため，本研究では難易度のスケールを小学 1 年，小学 4 年，小学 5 年，中学 3 年の 4 段階とする．

各難易度の差異率を表 8 に示す．

表 8. 各難易度の差異率

難易度	差異率（%）
小学 1 年	0〜2.8
小学 4 年	2.9〜5.7
小学 5 年	5.8〜8.6
中学 3 年	8.7〜11.5

表 8 のように差異率を定めたのは，小学 1 年では差

異率が 0～3.2%の範囲で一番多くデータが取れていたが，小学 4 年では差異率が 2.9～5.8%の範囲で一番多くデータが取れていた．そのため，小学 1 年の差異率の範囲を 0～2.8%にした．以下の難易度も同様にして定めた．

4. 評価実験と考察

本章では，評価実験の手法とその実験結果について述べる．さらに，その結果を考察する．具体的には評価データを用いた評価実験の結果と考察について述べる．

提案手法の精度を評価するために，小学 1 年，小学 4 年，小学 5 年，中学 3 年の教科書に載っている作品に対して各学年 20 個の文章を評価データとして集めた．評価実験の方法は，合計 80 個の文章に対して差異率の測定を行い，推定された難易度が実際の学年と一致するかを調べた．具体的には，推定された難易度が実際の学年と一致した場合の個数を集計し，割合を求めた．

提案手法の精度を評価するために，小学 1 年，小学 4 年，小学 5 年，中学 3 年の教科書に載っている作品に対して各学年 20 個の文章を評価データとして評価実験を行った．その結果を表 9 に示す．

表 9．各難易度の評価実験結果

難易度	精度
小学 1 年	55%
小学 4 年	55%
小学 5 年	45%
中学 3 年	45%
全体	50%

このような結果になった要因としては，各学年に対するデータの散らばりと平均値との差が大きいことが考えられる．

各学年のデータの散らばりを図 9，10，11，12 に示す．

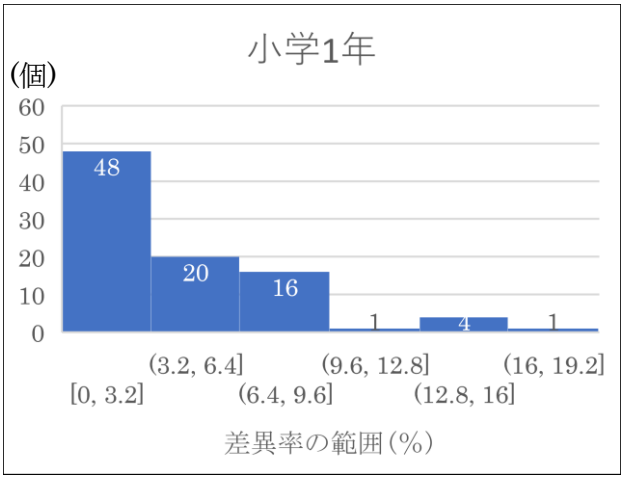


図 9．小学 1 年の差異率の分布

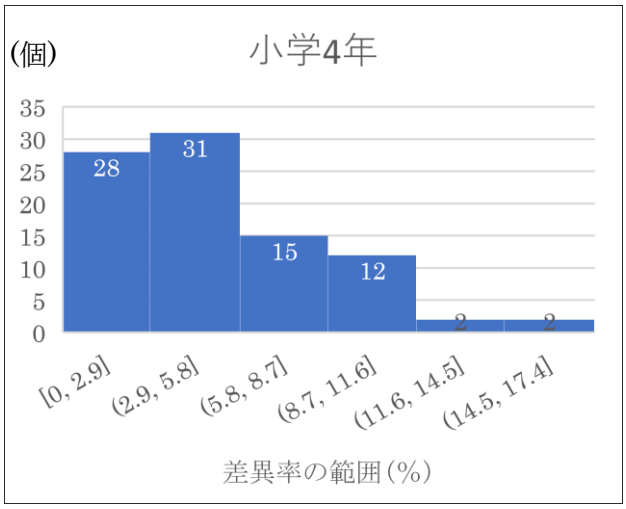


図 10．小学 4 年の差異率の分布

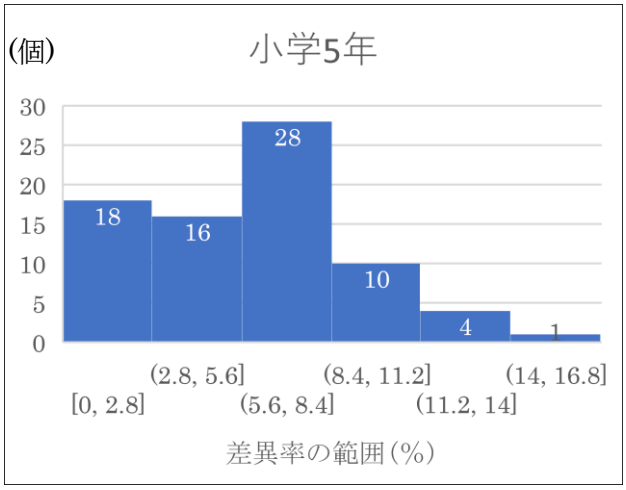


図 11．小学 5 年の差異率の分布

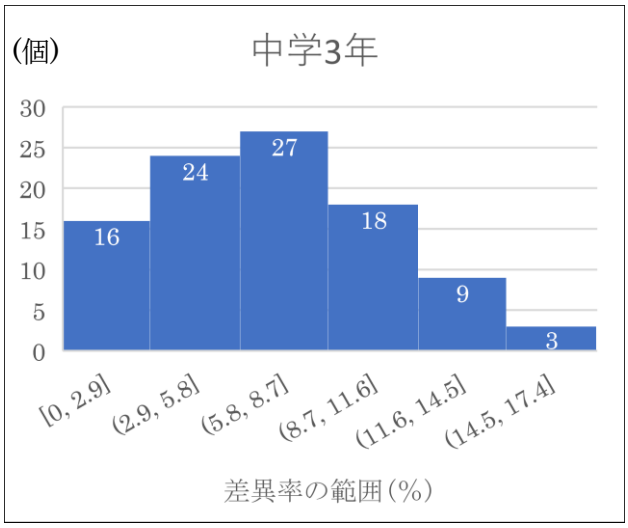


図 12．中学 3 年の差異率の分布

各学年の差異率の平均を表 10 に示す．

表 10. 各難易度と平均値

難易度	平均値	標準偏差
小学 1 年	4.186	4.039
小学 4 年	5.243	3.563
小学 5 年	5.849	3.343
中学 3 年	6.99	3.752

図 9 と表 10 より，分布で最も偏りのある 0〜3.2 の範囲を各難易に対する差異率として定義したが，平均値 4.186 はそれを外れている．また，図 10 から，2.9〜5.8 の範囲では 31 個，0〜2.9 の範囲では 28 個であるように分布で最も偏りのある範囲に対する個数があまり変わらないことも，表 9 の結果として考えられる．

そこで，相関の高かった小学 1 年と中学 3 年の 2 つのレベルを難易度のスケールとして差異率を定めた．その結果を表 11 に示す．

表 11. 各難易度の差異率

難易度	差異率（%）
小学 1 年	0〜5.7
中学 3 年	5.8〜11.5

この定めた差異率に対して評価実験を行った結果を表 12 に示す．

表 12. 各難易度の評価実験結果

難易度	精度
小学 1 年	80%
中学 3 年	65%
全体	72.5%

この結果より，4 つのレベルに対しての差異率の評価実験より精度が上がったことがわかる．

5. 今後の課題

本研究の今後の課題について以下に述べる．

（1）2 章で述べたひらがなやカタカナ，漢字の表記が異なる場合に形態素解析の読みで比較した部分については，漢字の読みが形態素解析では不十分であることがわかった．そのため，本研究では読みで diff を行った部分については手動で行うこととなった．理由としては，漢字の読みが文章中における正しい読みではない場合があることが原因である．この解決方法としては，漢字の読みを複数返してくれる技術を用いることが考えられる．これにより，形態素解析では 1 つの読みでしか 2 つの文章に対して diff が行えなかったが，複数の読みでの diff を行うことで一致する読みが見つかる可能性がある．読みに対する diff を手動で行った差異率と，自動での差異率との間に誤差が生まれた．具体的には，38%の相対誤差が生まれた．今後の課題としては，自動での差異率で難易度推定を行えることである．

（2）本研究では，難易度推定を小学 1 年，小学 4 年，小学 5 年，中学 3 年の 4 つで行なったが 50%の精度となった．そこで小学 1 年と中学 3 年の 2 つでおこなったところ 72.5%の精度で当初予定していた精度より低かった．今後の課題として，他の学年での集めた文章をもう 1 度精査することで 9 つのレベルの難易度推定を行う．

（3）本研究では，国語の教科書に載っている作品に対しての難易度推定を行ったが，教科書コーパスや均衡コーパスを用いることで幅広い分野に対しての差異率を調べる．

6. 結言

本論文では，日本語文章の読みやすさ（音読のしやすさ），聞き取りやすさを指標にして日本語文章の可読性を推定する手法の提案を行った．

小学 1 年生，小学 4 年生，小学 5 年生，中学 3 年生で習う各難易度の文章に対しての難易度推定を行った．小学 1 年から 6 年，中学 1 年から 3 年の各レベルにおける差異率の相関係数は 0.186 となり，ほとんど相関がないことがわかった．この結果から 9 つのレベルに対しての難易度推定は行えないことがわかった．そこで，小学 1 年生，小学 4 年生，小学 5 年生，中学 3 年生における差異率の相関係数を調べた結果，0.431 で相関があることがわかった．この結果から，4 つのレベルに対しては難易度推定に利用できる可能性が示された．さらに，小学 1 年生，小学 4 年生，小学 5 年生，中学 3 年生の各難易度に対する差異率を求めた．その差異率をもとに，4 つのレベルに対する本研究の提案手法を調べるために評価実験を行った結果，精度が 50%であることがわかった．この結果から，音声反訳による差違率のみで日本語の文章難易度推定を行なうことは困難であることが明らかになった．そこで，小学 1 年生，中学 3 年生における差異率の相関係数を調べた結果，0.533 で 4 つのレベルよりも相関があることがわかった．2 つのレベルの評価実験を行った結果，精度が 72.5%であることがわかった．このことから，2 つのレベルに対しては難易度推定に利用できる可能性が示された．

この問題の解決方法としては，集めた文章を精査することで各レベルの差異率のばらつきを小さくすることが考えられる．

本研究では国語の教科書に載っていた作品を集めたため，他の分野の文章での差異率はわからない．今後の課題として，より幅広い表現を用いた文章を集め，差異率を調べることが考えられる．

謝辞

本研究に際して，様々なご指導を頂きました西村俊二講師に深謝いたします．また，この研究の機会をくださった情報工学科の先生方，そして多くの知識やご指摘を下さいました同研究室の先輩・同期の皆様に厚く御礼申し上げます．

参考文献

[1] R. Flesch, “A new readability yardstick,” Journal of Applied Psychology,” Vol. 32. pp.221-233 , 1984.

[2] “日本語テキストの難易度を測る(帯 3),” [オンライン]. Available: http://kotoba.nuee.nagoya-u.ac.jp/sc/obi3/. [アクセス日: 20 10 2019].

[3] “日本語文章難易度判別システム(jReadability),” [オンライン]. Available: http://jreadability.net/. [アクセス日: 20 10 2019].

[4] 佐藤理史, “均衡コーパスを規範とするテキスト難易度測定,” 情報処理学会論文誌, Vol.52, No.4, p.1777-1789,, 2011.

[5] 李在鎬, “日本語教育のための 文章難易度に関する研究,” 早稲田日本語教育学 第 21 号 p.1-16（2016）, 2016.

[6] 赤. 信. 一宏, “リーダビリティ指標を用いた文章評価システムの開発： 計算機と大学生による可読性評価の比較,” 情報処理学会 第 15 回情報科学技術フォーラム(FIT2016)講演論文集 pp.293-294, 2016.

[7] W3C, “Web speech api specification,” 2019. [オンライン]. Available: https://wicg.github.io/speech-api/. [アクセス日: 7 1 2020].

[8] MDN, “Web Speech API,” 23 3 2019. [オンライン]. Available: https://developer.mozilla.org/ja/docs/Web/API/Web_Speech_API. [アクセス日: 7 1 2020].

[9] aizulab, “Web Speech API の音声認識と音声合成を使ってみる,” 2019. [オンライン]. Available: http://www.aizulab.com/blog/try-using-the-web-speech-api/. [アクセス日: 7 1 2020].

[10] so-zou, “WebSpeechAPI,” [オンライン]. Available: https://so-zou.jp/web-app/tech/programming/javascript/media/audio/web-speech-api/. [アクセス日: 7 1 2020].

[11] K. Decker, “kpdecker/jsdiff (online),” [オンライン]. Available: https://github.com/kpdecker/jsdiff. [アクセス日: 7 1 2020].

[12] E. W. Myers, “An O(ND) Difference Algorithm and Its Variations,” Algorithmica, 1:251- 266, 1986, 1986.

[13] TakuyaAsano, “takuyaa/kuromoji.js,” [オンライン]. Available: https://github.com/takuyaa/kuromoji.js/blob/master/README.md. [アクセス日: 7 1 2020].

[14] ZENY, “kuromoji.js を使って形態素解析,” [オンライン].Available: https://zeny.io/blog/2016/06/16/kuromoji-js/. [アクセス日: 7 1 2020].

[15] MDN, “正規表現,” [オンライン]. Available: https://developer.mozilla.org/ja/docs/Web/JavaScript/Guide/Regular_Expressions. [アクセス日: 7 1 2020].

[16] pisuke-code, “JavaScript でひらがな・カタカタ・漢字をチェックする方法まとめ,” [オンライン]. Available: https://pisuke-code.com/js-check-hira-kana-kanzi/. [アクセス日: 7 1 2020].

[17] マサト, “replace の文字列置換・正規表現の使い方まとめ,” [オンライン]. Available: https://www.sejuku.net/blog/21107. [アクセス日: 7 1 2020].