特別研究報告書

折り返し翻訳を用いた自己主導型リペア支援

指導教員　　石田亨 教授

京都大学工学部情報学科

宮崎真介

平成17年 2月10日

折り返し翻訳を用いた自己主導型リペア支援

宮崎真介

内容梗概

近年のインターネットの急速な広がりに伴い，国境を越えた大容量の情報交換が安価で可能になってきている．また，多くの日系企業が安価な労働力を大量に確保するという目的で，オフショア開発によるソフトウェアの開発を積極的に行っている．しかし，このオフショア開発は言語の異なるメンバーから構成されるために，スタッフの間でのコミュニケーション不足が原因でトラブルも起きている．

このようなコミュニケーションの問題に取り組む形で，本研究室で 2002 年に，ICE2002 という実験が行われた．ICE2002 は，日中韓馬からの参加者がオンライン上でディスカッションをしながら，オープンソースソフトウェアの共同開発を行うというものであり，参加者は機械翻訳を利用することで自分の母国語でディスカッションに参加した．この実験の中で何回も機械翻訳を介したコミュニケーション (ディスカッション) が行われたが，機械翻訳はコミュニケーション支援のために作られたツールではないために，参加者同士がコミュニケーションを行うのに十分な翻訳品質を提供できなかった．このため，参加者は自分の入力した文 (原文) が機械翻訳に適したものになるように，原文の英訳を参考にして何度も原文を書き直すことを強いられた．しかし，目標言語の知識が乏しい参加者にとっては，自分の原文がうまく翻訳されているかの判断もできない上に，原文の中でどの部分が機械翻訳に適していないのかわからないために，非常に効率の悪い書き直しを強いられた．ICE2002 のこのような結果から，以下の問題が見つけられた．

- 機械翻訳がうまく翻訳できないときに，原文のどの部分がその原因の箇所であるかがわからない．

この問題を解決するために，自己主導型の機械翻訳支援ツール SimTrans の開発を行った．SimTrans はユーザからの原文をもとに，その原文が機械翻訳によってうまく翻訳されるかを判断し，うまく翻訳されないときには，原文の中で機械翻訳に不適な部分をユーザに指摘するツールである．SimTrans はその計算過程で，折り返し翻訳 (Back-Translation) と類似度計算ツールと文短縮ツールを利用した．折り返し翻訳とは，原文を目標言語に翻訳し，その翻訳結果を

原言語にもう一度翻訳することである．類似度計算ツールとは，２つの文がどれだけ似ているかを計算するツールである．文短縮ツールとは，文を文節レベルまで分解し，その全文節の中からいくつかの文節を取り出して，意味の通る短い文を生成するツールである．SimTrans はこれらのツールを利用して，原文の機械翻訳に適していない部分を以下のような手法で検出し，指摘する．

SimTrans は，まず原文とその折り返し翻訳結果との類似度を計算し，十分に大きければ，うまく翻訳されると判断し，そうでなければ，うまく翻訳されないと判断する．SimTrans が「うまく翻訳されない」と判断した際には，そう判断された原因となる部分の検出に入る．そのために，まず原文の短縮文を生成する．生成された複数の短縮文ひとつひとつに対して，うまく翻訳されるかを，折り返し翻訳と類似度計算ツールを利用して判断する．その結果から,「うまく翻訳される短縮文」に含まれていない文節を，原文がうまく翻訳されない原因部分であると見なして，そのような文節に対してスコアを付けていく．最終的に，原文を構成する文節の中で最もスコアが高くなった文節を，機械翻訳に適していない部分として，ユーザに指摘する．

このような機能に加え，短縮文ひとつひとつについての類似度や翻訳結果を SimTrans がどのような過程を経て処理しているのかを詳細に確認できるデバッグモードや，SimTrans が処理した短縮文・類似度・翻訳結果などを全て XML によるログデータで保存する機能などを実装することによって，原文のどの部分が機械翻訳に不適切であるのかを解析できるデータを出力するようにした．

最後に SimTrans の効果を検証するために，機械翻訳に適していない部分を含む 64 個の例文を SimTrans に適用した．その結果として，以下のような成果が得られた．

1. 文の翻訳不適箇所に，文節単独で翻訳しにくい文節が含まれているものは，機械翻訳不適箇所として指摘されやすい．
2. 文末の翻訳不適箇所は指摘されやすい．
3. 折り返し翻訳や類似度計算の値が不適切になると SimTrans はそれに対応できない．
4. 文中に問題のある部分があっても，その部分を含んだ短縮文の短-折類似度が高くなってしまうと対応できない．

Supporting Self-Initiated Repair Using Back-Translation

Shinsuke MIYAZAKI

Abstract

Abstract The rapid spread of Internet in recent years has enabled us to easily exchange large amount of information across borders. Also for the purpose of securing cheap manpower in large quantities (mass labor), many Japanese enterprises are developing software through offshore development. However, because this offshore development is advanced by people using different languages, troubles are caused by lack of the communication between the staff.

To tackle these problems, Intercultural Collaboration Experiment was carried out in 2002. In this experiment, the participants repaired their text many times to make the sentences suitable for machine translation (MT). But a participant who did not have enough knowledge of the target language had low efficiency in rewriting because they could not judge if the sentences were translated well or which parts of the sentence were not fit for MT. Based on the ICE2002 experimental results, we found the following problems.

When the MT can not translate well, the user does not know which part of the sentence is the cause. To solve this problem, we developed a self-initiated repair system (SimTrans).

Our laboratory made an experiment called ICE2002 in 2002. ICE2002 is a joint development of open source software experiment conducted by people from Japan, China, Korea, and Malaysia. Discussions between members were carried out over the Internet. The participants used their native language to communicate with each other by relying on MT for support. However, limited discussions were carried out using the MT, since the tool was not designed to support interactive communication. Moreover, MT tool could not provide enough quality translation to carry out smooth communication. Due to this, the participant had to repair the text they wrote so that it is suitable for MT.

SimTrans judges if the text will be translated well. SimTrans receives sentences created by the user, and if the text is not suitable for machine translation, it points out the unsuitable parts.

Simtrans uses back-translation, similarity measuring tool and sentence ab-

breviation tool during calculation. Back-translation translates the text first into the target language and then it translates that result back in to the participant's own language. The similarity measurer calculates how similar the two sentences are. Sentence abbreviator breaks sentences into segments and picks up some segment from all of them to create short sentences which make sense. Simtrans uses these tools to point out the unsuitable parts from the sentences.

SimTrans calculates the similarity between the original sentence and back-translated sentence. If the similarity is large enough, SimTrans judges that original sentence is translated well. Otherwise, the SimTrans judges that the original sentence is not suitable for translation. When the original sentence is not translated well, SimTrans attempts to find the part that cause the problem in the sentence. To find the cause, SimTrans creates abbreviated sentences first. Second, it judges if MT can translate each of the abbreviated sentences (short sentences) (broken down sentences) well or not by using the back-translation and similarity measuring tool . In all segments of the original sentence, SimTrans points to the segments which are not included as well-translated sentence. SimTrans points out the segment, which were given the highest score, to the user as the problematic part. In addition to such functions, DebugMode and XML-LogData were implemented. DebugMode and XML-LogData is designed to trace how SimTrans process the translation-results and similarity of each abbreviated sentences. The user in turn can also analyze where the most inappropriate part of the sentence is by using these functions.

As the end of this research, SimTrans was applied to 64 sentences each of which has the parts unsuitable for MT. The results are following.

1. The unsuitable parts which were often pointed out were the segments which are too difficult to translate by MT.
2. It was easy to point out the unsuitable parts in sentence end.
3. SimTrans can not accommodate to the conditions when the result of back-translation or similarity measuring were inappropriate.
4. Even if unsuitable parts were included in the sentence, it is impossible for SimTrans to point out them if the abbreviated sentences which include them are appropriate to MT.

折り返し翻訳を用いた自己主導型リペア支援

目次

第1章　はじめに

近年，アジア各地でインターネットの利用が急速に広まっている．利用人口が増えているだけでなく，通信できる情報の量が増えているために，オンライン上での異文化間コラボレーション環境へのニーズが高まっている．また，日本の多くの企業が，安価な労働力を大量に得られるオフショア開発をインドや中国の企業に積極的に委託している状況からも，アジアでの多国籍プロジェクトが今後さらに増えていくと考えられる．その一方で，アジアでの多国籍プロジェクトには，文化・言語・民族などが異なる様々な参加者がいるために，コミュニケーションに関する問題も発生する．オフショア開発においても，主に言葉の違いから来るスタッフ間のコミュニケーション不足が原因で納期や品質に関するトラブルが起きることが報告されている．

こういった言葉の問題は日本を含む東アジア諸国においては特によく発生する．なぜなら，ヨーロッパ諸国の言語のように近くの国の言語が自分の国の言語と似ているわけではないことに加え，隣の国の言葉を教育される機会が無いためである．しかし，アジアの多くの人々にとって，世界共通語とされている英語を用いて，高度で，論理的な内容を含むコミュニケーションを行うことは難しい．

このアジアの国々の国境を越えたコミュニケーションに関する問題に取り組む形で行われたのが，2002 年度に本研究室で行われた異文化コラボレーション実験 2002（ICE2002）[1] [2] である．この実験は，日中韓馬の 4 カ国間でソフトウェアの開発を協力して行うというものであったが，各種の議論は機械翻訳を介することによってそれぞれの参加者の母国語で行われた．この実験から以下のことが報告されている。

- 機械翻訳システムはコミュニケーションを支援するために設計されたものではないために，その翻訳品質はコミュニケーションを支援するのに十分なものではなかった．
- 参加者は自分の入力した文 (原文) がうまく翻訳されるように，試行錯誤して原文を何度も書き直したが，機械翻訳に適した文を書くことは困難であった．

ICE2002 の上記のような実験結果から，機械翻訳を介したコミュニケーションに関して以下の様な問題点が見つけられた．

- 機械翻訳がうまく翻訳していない際に，原文のどの部分が原因で機械翻訳がうまくいかないのかがわからない．

この問題点を解決するために，ユーザが入力した文の中でどの部分が翻訳に適していないのかを指摘するツールSimTransの開発を行った．ユーザはSimTransに翻訳して欲しい文を入力し，SimTransは機械翻訳に不適な箇所を検出し，その部分を指摘するメッセージをユーザに送る．SimTransからのメッセージを受け取ったユーザは，そのメッセージをもとに原文をリペアし，新しい原文として，改めてSimTransに入力する．この作業の繰り返しによって原文をリペアしていくことで，単純に原文をリペアするよりも効率よくリペアできるようにすることをSimTransの目的とする．

第2章　自己主導型リペア支援システムSimTrans

2.1　自己主導型リペア支援

2.1.1　異文化コラボレーション実験 (ICE2002)

アジア地域での高速インターネットの急速な普及に伴い，インターネットを利用したシステムへの期待は高まり，オンライン上での異文化コラボレーション環境も必要とされている．また，日本のソフトウェア開発においても，安くて大量の労働力を目当てに，中国やインドの企業にオフショア開発を積極的に委託している．その一方で，こういった多国籍なプロジェクトは問題も抱えている．言語や文化の異なる参加者が共同作業を行うために，コミュニケーションがうまく取れず，そのことが共同作業の妨げとなることである．オフショア開発に関しても，現地で採用したスタッフが十分な技術を持っていなかったり、主に言葉の違いから来るスタッフ間のコミュニケーション不足が原因で納期や品質に関するトラブルが起きたりしている．

本研究室で2002年に行われた異文化コラボレーション実験 (ICE2002) とは，上記のような，言語や文化が異なる参加者の共同作業におけるコミュニケーションの問題に取り組む形で行われた実験である．ICE2002では，日本・中国・韓国・マレーシアの大学からの参加者たちによって，ディスカッションをしながらのソフトウェア開発が行われた．その際に，母国語がそれぞれ異なる参加者がそれぞれの母国語でディスカッションできるために，機械翻訳を介してディスカッションが行われた．この実験結果として以下のことが報告されている．

- 機械翻訳システムはコミュニケーションを支援するために設計されたものではないために，その翻訳品質はコミュニケーションを支援するのに十分なものではなかった．
- 参加者は自分の入力した文 (原文) がうまく翻訳されるように，試行錯誤して原文を何度も書き直したが，機械翻訳に適した文を書くことは困難であった．ある参加者においては，ひとつのメッセージを送信するの約 8 回ものリペアを行っていた．

この実験結果から，機械翻訳を介したコミュニケーションにおける以下の様な問題点が見つかった．

- 機械翻訳がうまく翻訳していない際に，原文のどの部分が原因で機械翻訳がうまくいかないのかがわからない．

2.1.2 自己主導型リペア支援システムの導入

ICE2002 では前節で説明したとおり，機械翻訳の翻訳結果に関する問題が見つかっている．その問題点の解決のために，以下に説明するようなシステムを設計・実装する．

このシステムは，機械翻訳を利用したユーザがその翻訳結果をリペアすることを支援する [3] ことを目的とするものである．具体的には，以下のような処理を繰り返すことでリペアが効率よく行われるようにする．

1. ユーザが入力した文 (原文) を受け付ける．
2. 原文が機械翻訳によってうまく翻訳されているかを判断し，うまく翻訳されていないと判断した際には，その原因となる部分を検出する．これらの結果をユーザに伝える．
3. ユーザの入力を待つ．

システムのこのような動きに対してユーザは図 2.1 のようにリペアを繰り返すことで，機械翻訳に適した文を作成していく．

なお，このシステムでは，システムから指摘される内容をユーザが自分で判断してリペアを行うため，このシステムを自己主導型機械翻訳支援システムと呼ぶこととする．

また，どのようにしてうまく翻訳されない原因の部分を検出するかは後述する．

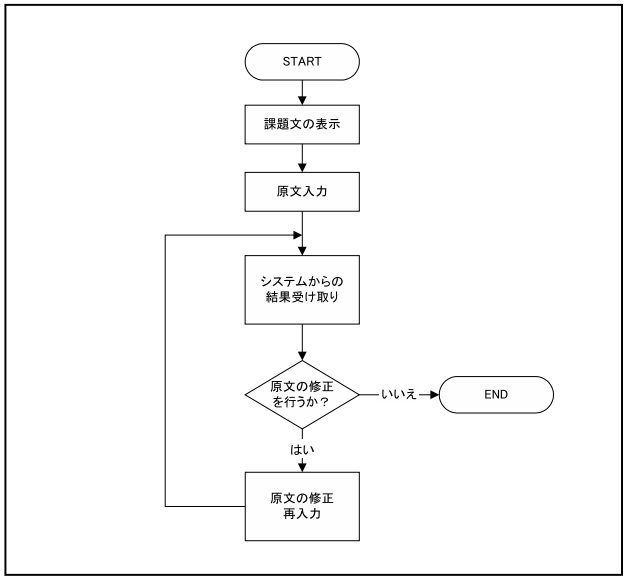

図 2.1: リペアの流れ

2.2 機械翻訳不適箇所の検出

本節では，機械翻訳がうまく翻訳できない原因となる部分 (機械翻訳不適箇所) を，すでに説明した自己主導型機械翻訳支援システムがどのようにして検出するかを述べる．

2.2.1 機械翻訳不適箇所の検出に使われるツールと手法

まず，本システムが機械翻訳不適箇所を検出する際に用いるツールや手法の説明をする．本システムでは以下に挙げる 3 つを利用して，翻訳不適箇所を指摘する．

1. 折り返し翻訳 (Back-Translation)
2. 類似度計算ツール
3. 文短縮ツール

折り返し翻訳とは表 2.1 のように，原文を目標とする言語 (目標言語) へと翻訳した上で，その翻訳結果の文をもう一度原文と同じ言語 (原言語) に翻訳することである．また，理想的な機械翻訳であれば，原文と折り返し翻訳の結果の文は同じ文になる．なお，これ以降，折り返し翻訳の結果としてできた原言語の文を折り返し文と呼ぶ．

類似度計算ツールとは，２つの文を入力として受け取り，その 2 つの文がどれだけ似ているかを計算し，0 から 1 までの類似度 [2] として表す．今回の実験で用いた類似度計算ツールは，6 種類の類似度が返ってくるので，類似度が必

表 2.1: 折り返し翻訳

文の種類	言語の種類	文
原文	原言語 (日本語)	私は，明日から旅行に行く予定です。
訳文	目標言語 (英語)	I plan to take a trip from tomorrow.
折り返し文	原言語 (日本語)	私は、明日から旅行をすることを計画します。

要な際にはそれらの中から適切なものを選んだ．なお，この類似度計算ツールは，本研究を共同で行っている NICT に提供していただいた。

文短縮ツールとは，ひとつの文を入力として受け取り，短縮文の集合を出力するツールである．短縮文とは，入力された文の構文解析結果をもとに作られた，入力された文よりも短い文である．例えば文短縮ツールへの入力として,「私は、明日から旅行に行く予定です。」という文が与えられたとする．このとき，この文の構文解析結果は図 2.2 のようになる．ただし，図 2.2 を出力するのに，日本語構文解析システム KNP を用いた．

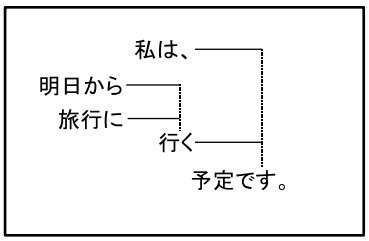

図 2.2: 「私は、明日から旅行に行く予定です。」の構文解析結果

図 2.2 の構文解析結果から生成された短縮文の集合を表 2.2 に示す.

ただし，表 2.2 では，入力された文そのもの (表 2.2 中の 17 番) も短縮文となっているが，今回のツールでは入力された文そのものは短縮文の集合から除いて処理をした.

なお，このツールに関しても類似度計算ツールと同じく NICT に提供していただいた.

2.2.2 機械翻訳結果の判断

本システムは機械翻訳に適していない文の中で機械翻訳不適切箇所を検出する．つまり，文が機械翻訳に適しているかの判断を行う部分が存在する．ここでは，本システムがどのようにして，入力された文を機械翻訳に適しているか判断しているかを，具体例を用いて説明する.

表 2.2: 「私は、明日から旅行に行く予定です。」から生成された短縮文の集合 (NICT 提供の文短縮ツールの出力による)

番号	短縮文
1	私は、
2	明日から
3	旅行に
4	行く
5	明日から 行く
6	旅行に 行く
7	明日から 旅行に 行く
8	予定です。
9	私は、 予定です。
10	行く 予定です。
11	明日から 行く 予定です。
12	旅行に 行く 予定です。
13	明日から 旅行に 行く 予定です。
14	私は、 行く 予定です。
15	私は、 明日から 行く 予定です。
16	私は、 旅行に 行く 予定です。
17	私は、 明日から 旅行に 行く 予定です。

例えば,「幸運に恵まれて、私は大学に入りました。」という文が原文であった場合，機械翻訳にかけた結果として,「Fortunately, it was fortunate and I entered a university.」という英文を翻訳結果として受け取る．この翻訳結果をさらにもう一度機械翻訳にかけて,「幸運にも、それは幸運で、私は大学に入りました。」という折り返し文を得る (表 2.3 参照).

表 2.3: 折り返し翻訳例

文の種類	文
原文	幸運に恵まれて、私は大学に入りました。
英訳	Fortunately, it was fortunate and I entered a university.
折り返し文	幸運にも、それは幸運で、私は大学に入りました。

次に，原文と折り返し文との間の類似度を，類似度計算ツールによって計算する．その類似度が一定の値を超えていれば，入力された原文は「うまく翻訳された」と判断し，一定の値に満たなければ「うまく翻訳されなかった」と判断する．このようにして，入力された文が機械翻訳に適しているかを判断する．

今回の例では，6 種類の類似度はそれぞれ表 2.4 のとおりであったが類似度の

しきい値を 0.5 とし，類似度のタイプを F 値としてあったとすれば,「うまく翻訳されていない」と判断される．

表 2.4: 類似度例 (しきい値:0.5)

(NICT 提供の類似度計算ツールによる)

計算手法	precision	recall	F 値
置き換えあり	0.568436743985199	0.299971754534595	0.392706814191284
置き換えなし	0.568436743985199	0.299971754534595	0.392706814191284

2.2.3 機械翻訳不適箇所の検出アルゴリズム

次に，本システムが機械翻訳不適箇所をどのように検出するかを述べる．まず，本システムが用いる機械翻訳不適箇所検出のアルゴリズムを以下に示す．

1. **問題定義**

   文 S を構成する文節の中で最も翻訳に適さない文節 seg_{bad} を求める．ここにおける「最も翻訳に適さない文節」とは，seg_{bad} と S を構成する任意の文節 $seg_x(\neq seg_{bad})$ とで重み W を比較したときに，$\mathrm{W}[seg_{bad}] \geq \mathrm{W}[seg_x]$ という条件を満たすような文節 seg_{bad} のことを指す．重み W については後で定義する．

2. **変数・関数定義**

   変数

   - S は文である．
   - abr_{temp} は短縮文である．ただし，短縮文とは文 S が与えられたとき，その構文解結果から得られる文 S よりも短い文である．
   - ABR は短縮文の集合である．
   - seg_{temp} は文節である．
   - C はしきい値であり，[0,1] の定数である．
   - W[x] は x の重みであり，0 以上の正数値である．x は文節または短縮文である．
   - ABR_M は，$\forall seg_{temp}(\in ABR_M)$ について，$\mathrm{W}[seg_{temp}] > \mathrm{C}$ を満たすような短縮文の集合である．
   - $W_{abr}[\ ]$ は配列である．$W_{abr}[abr_i]$ には短縮文 abr_i の重み $\mathrm{W}[abr_i]$ が入る．
   - $W_{seg}[\ ]$ は配列である．$W_{seg}[seg_i]$ には文節 seg_i の重み $\mathrm{W}[seg_i]$ が入る．

関数

- **RTRANS(S)** は，文 S を折り返し翻訳した結果の文を返す
- **SIMILARITY**(S_1,S_2) は，2 つの文 S_1 と S_2 の間の類似度を返す.
- **ABBREVIATE**(S) は，文 S から生成された短縮文の集合を返す.
- **DIFF**(S_1,S_2) は，文 S_1 を構成する文節集合と文 S_2 を構成する文節集合の差分集合を返す.
- **WEIGHT**(ABR_M) は短縮文の集合 ABR_M の要素ひとつひとつに対して重みを計算し，要素と与えられた重みを対応させた配列を返す.
- **K(seg)** は文節 seg の指摘率 (式 2.1 参照) を返す.
- **MAX**(W_{seg}) は，$W_{seg}[x] \in W_{seg}, W_{seg}[y] \in W_{seg}$ $(W_{seg}[y] \neq W_{seg}[y])$ について， $\exists W_{seg}[x] \geq \forall W_{seg}[y]$ を満たすすべての x を返す.

3. **手順**

(i) ABR $\leftarrow$ ABBREVIATE(S)

(ii) **for each** abr_{temp} **in** ABR **do**
　　if(SIMILARITY(abr_{temp},RTRANS(abr_{temp})) $>$ C **) then**
　　$ABR_M \leftarrow ABR_M \cup abr_{temp}$
　　end of if
　end of for

(iii) $W_{abr} \leftarrow$ WEIGHT(ABR_M)

(iv) **for each** abr_{temp} **in** ABR_M **do**
　　for each seg_{temp} **in** DIFF(S,abr_{temp}) **do**
　　$W_{seg}[seg_{temp}] \leftarrow W_{abr}[seg_{temp}] \times$ K$[seg_{temp}]$
　　end of for
　end of for

(v) **return** MAX(W_{seg})

以下では具体例を用いて，上記のアルゴリズムを説明する．2.2.2 で述べた処理で，原文の機械翻訳の結果が「うまく翻訳されている」と判断されれば，機械翻訳不適箇所を特定する必要はないが,「うまく翻訳されていない」と判断されたときには，機械翻訳不適箇所を検出する処理に移る．ここで先に述べたアルゴリズムが用いられる.

機械翻訳不適箇所を検出するために，まず，ステップ　として原文から短縮文の集合を生成する．今回の例文では，表 2.5 のようになる．この出力には NICT 提供の文短縮ツールを利用した．

表 2.5: 「幸運に恵まれて、私は大学に入りました。」の短縮文の集合

(NICT 提供の文短縮ツールの出力による)

番号	短縮文
1	幸運に
2	恵まれて、
3	幸運に 恵まれて、
4	私は
5	大学に
6	入りました。
7	恵まれて 入りました。
8	幸運に 恵まれて、 入りました。
9	私は 入りました。
10	恵まれて、 私は 入りました。
11	幸運に 恵まれて、 私は 入りました。
12	大学に 入りました。
13	恵まれて 大学に 入りました。
14	幸運に 恵まれて、 大学に 入りました。
15	私は 大学に 入りました。
16	恵まれて、 私は 大学に 入りました。
17	幸運に 恵まれて、 私は 大学に 入りました。

次のステップ　として，得られた短縮文の集合から機械翻訳に適しているものを取り出す．そのために，短-折類似度を用いる．短-折類似度とは，短縮文集合に属するひとつの短縮文を原文とした，原文と折り返し文との間の類似度のことである (図 2.3 参照)．

得られた短縮文ひとつひとつについて，短-折類似度を計算する．これらの中から，類似度の値が一定以上の値 (アルゴリズムの中の C にあたる値) を超えた短縮文を取り出し，これらを機械翻訳に適した短縮文とする．今回の例文の場合，機械翻訳に適した短縮文は表 2.6 のようになる．

次にステップ　として，これら機械翻訳に適した短縮文の中でどの短縮文を最も重要とするかを決めるのだが，重要さを決めるために，このソートに用いるキーが必要となる．今回のシステムでは優先する順に以下の 3 つをキーとして用いた．

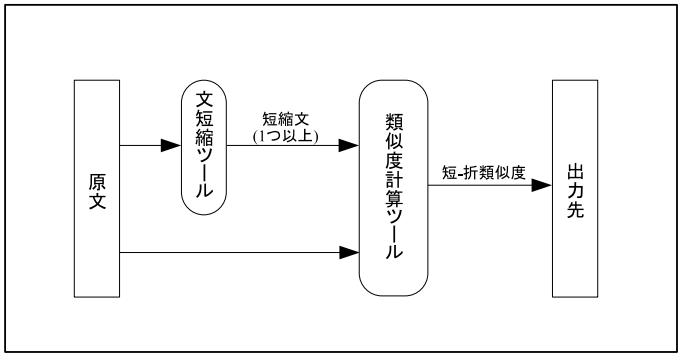

図 2.3: 短-折類似度

表 2.6: 機械翻訳に適した短縮文の集合

番号	短縮文
6	入りました。
9	私は 入りました。
12	大学に 入りました。
15	私は 大学に 入りました。
16	恵まれて、 私は 大学に 入りました。

1. 原文類似度
2. 1st 原文類似度
3. 短-折類似度

原文類似度とは，短縮文と原文の間の類似度である．1st 原文類似度とは 1 回目に入力された原文と短縮文との間の類似度である．このシステムでは，出力結果をもとにユーザが原文をリペアして行くので複数回の原文入力とそれに伴う原文の変化が想定されている．そこで，1 回目の原文を他の原文と区別して評価することにしている．短-折類似度は既に述べたとおりである．また，今回の例における，機械翻訳に適した短縮文について，これら 3 つの値を表 2.7 に示す．これらの類似度の計算には NICT 提供の類似度計算ツールを用いた．

機械翻訳に適した短縮文の集合をソートした結果として，機械翻訳に適した短縮文が重要である順番に並ぶ．そして，ソートされた順にそれぞれの短縮文に重みを与える (表 2.8 参照)．ただし，今回の場合は順位が一つ違うと重みが 1.2 だけ差が出るようにした．これまでの，短縮文の集合要素ひとつひとつに順序を定めて，その順序に応じて重みを与える操作がアルゴリズム中の **WEIGHT** に相当する．

表 2.7: 機械翻訳に適した短縮文のキーごとの値

(NICT 提供の類似度計算ツールの出力による)

機械翻訳に適した短縮文	原文類似度	1st 原文類似度	短-折類似度
入りました。	0	0	1
私は入りました。	0.0469	0.0469	1
私は大学に入りました	0.2936	0.2936	1
大学に入りました。	0.0469	0.0469	1
恵まれて，私は大学に入りました.	0.7482	0.7482	0.6198

表 2.8: 短縮文への重み付け例

順位 [位]	番号	短縮文	重み付け
1	16	恵まれて，私は大学に入りました.	6.0
2	15	私は大学に入りました	4.8
3	9	私は入りました。	3.6
4	12	大学に入りました。	2.4
5	6	入りました。	1.2

さらにステップ　として，短縮文に与えられた重みを文節に反映させる．機械翻訳に適した短縮文に含まれていない文節 (15 番の短縮文なら「幸運に」と「恵まれて、」) に対して，その機械翻訳に適した短縮文が持つ重みを与えていくのだが，このままでは問題がある．

それは，それぞれの文節の合計出現回数と，被指摘回数 (機械翻訳に不適な文節だと指摘された回数) との関係を考慮していない点である．今回の例で言えば,「幸運に」の出現回数は 5 回であるのに対して,「入りました。」の出現回数は 11 回である．その一方で,「幸運に」の被指摘回数が 5 回であるのに対して,「入りました。」の被指摘回数は 0 回である (表 2.9 参照)．このように，原文から短縮文を生成したときに，ある文節が他の文節よりも多い回数だけ現れることが非常によくある．

ところで，短縮文の集合の中に多く現れるということは，翻訳に適した短縮文に含まれる可能性も高くなるということになる．そのため，出現回数に応じた処理を行うために，指摘率という値を導入する．指摘率の定義は以下の通りである．

$$\text{指摘率} = \frac{\text{文節 k が機械翻訳に不適であると指摘される回数}}{\text{短縮文集合中の文節 k の出現回数}} \tag{2.1}$$

この指摘率は定義の通り，文節によって定まる値である．対象の文節が短縮文の集合の中にたくさん出現すればするほど指摘率は小さくなる．また，対象の文節が機械翻訳に適した部分であると指摘される回数が多ければ多いほど指摘率は大きくなる．

この指摘率をそれぞれの文節に与えられる重みにかけたものを，文節に与えられる重みとする．こうすることで，短縮文の重みが，それぞれの文節の出現回数に応じてそれぞれの文節に与えられることになる．

この指摘率をかけた重みをそれぞれの文節に与え，全ての文節の中で最も大きな重みを与えられた文節が機械翻訳不適箇所として検出される．

以上が機械翻訳不適箇所を検出するアルゴリズムである．

最後になるが，表 2.9 に指摘率を考慮した場合とそうでない場合の，文節への重み付けの違いがどのように現れるかの例を示す．ただし，表中において，重み付け A とは指摘率を考慮しなかった場合の重み付け結果で，重み付け B とは指摘率を考慮した場合の重み付け結果である．

表 2.9: 文節への重み付け例

文節	出現回数	被指摘回数	指摘率	重み付け A	重み付け B
”幸運に”	5	5	1	19.2	19.2
”恵まれて”	9	4	4/9	13.2	5.87
”私は”	6	2	1/3	4.8	1.6
”大学に”	6	2	1/3	4.8	1.6
”入りました”	11	0	0	0	0

2.2.4 検出アルゴリズムの補助

ここまでのアルゴリズムで機械翻訳不適箇所が検出されない場合がある．その際の対応として，短-折類似度のしきい値 (アルゴリズム中の C) を小さくした上でもう一度上記のアルゴリズムを実行する．こうすることで，翻訳に適した短縮文だと判断される短縮文が増え，機械翻訳不適箇所が指摘される可能性が出てくる．このように，短-折類似度のしきい値を下げて検出アルゴリズムを実行することを，機械翻訳不適箇所が指摘されるか，それ以上しきい値が下げられない状態になるまで繰り返す．

第3章　SimTransの実装

これまでに説明した自己主導型リペア支援システムを実装したのが，今回のツールSimTransである．本章ではSimTransについて具体的な図を交えながらその機能を説明していく．

なお，SimTransの実装にはプログラミン言語C#を用いて行なった．また，今回使用した計算機の性能は表3.1のとおりである．

表3.1: 使用した計算機の性能

	マシン性能	
SimTrans	Pentium4	2.0GHz
外部翻訳サーバ	Pentium4	2.4GHz
文解析サーバ	Pentium4	3.0GHz

3.1　システム構成

まず，システム全体の構成を説明する．システム構成図を示すと，図3.1のようになる．

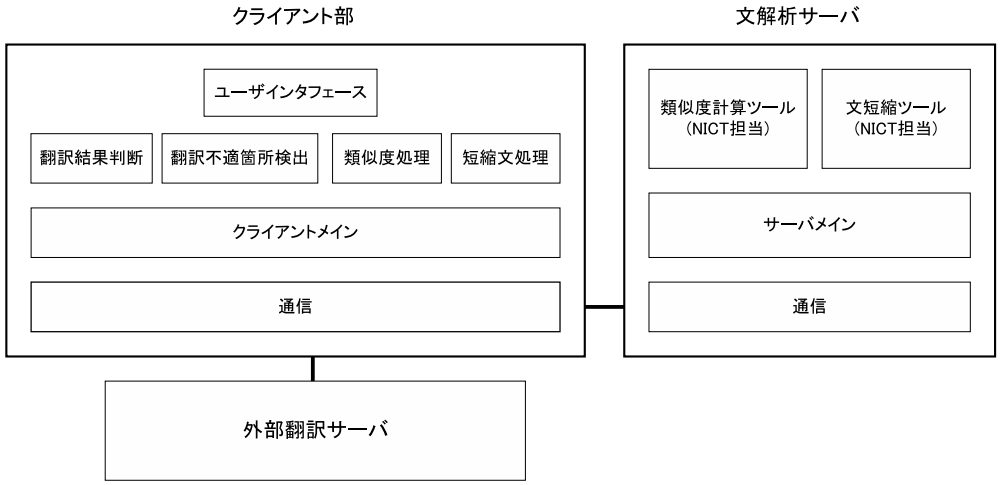

図3.1: システム構成図

- **ユーザ**

  図中にはないが，本システムの利用者である．クライアント部のユーザインタフェースを通じて本システムを利用する．

- **クライアント部**

  ユーザから入力された原文を直接受け付ける部分である．クライアント部では，通信部を通じて文解析サーバと通信を行い，文短縮ツールの計算結果を短縮文処理部で処理し，類似度計算ツールの計算結果を短縮文処理部で処理する．また，ユーザから入力された文が機械翻訳に適したものかを判断する翻訳結果判断部と，ユーザから入力された文が機械翻訳に適さなかった場合に機械翻訳不適箇所を検出する翻訳不適箇所検出部がある．これらの結果をユーザインタフェースを通じてユーザに伝える．なお，本研究は NICT との共同研究であるが，クライアント部は著者が担当をした．
- **外部翻訳サーバ**

  クライアント部から要求された翻訳を行いその結果を返す翻訳サーバである．
- **文解析サーバ**

  クライアント部から依頼された処理を行いその計算結果を返すサーバ．クライアント部から送られてくる要求は以下の 3 種類である．

  1. 2 つの文の間の類似度計算
  2. 文の短縮文の生成
  3. 文を構成する文節の取り出し

  1 については類似度計算ツールが計算結果を返し，2 と 3 については文短縮ツールが計算結果を返す．通信部分を通じて上記のように要求に応じた計算結果を返す．この類似度計算ツールと文短縮ツールについては共同研究を行った NICT に提供していただいた．

3.2 SimTrans のシーケンス図

次に SimTrans のそれぞれの部分の働きをシーケンス図を用いて説明する．

3.2.1 文解析サーバのシーケンス図

まず，文解析サーバがどのように動くかをシーケンス図を用いて説明する (図 3.2，図 3.3 参照)．

これらのツールは基本的に同じ動きをする．文解析サーバの通信部を通じて類似度計算を行うか文短縮を行うかの要求が送られてくる．この要求に対して，類似度計算ツールは 2 つにの文の間の類似度を返し，文短縮ツールは原文から生成できる短縮文の集合を返す．

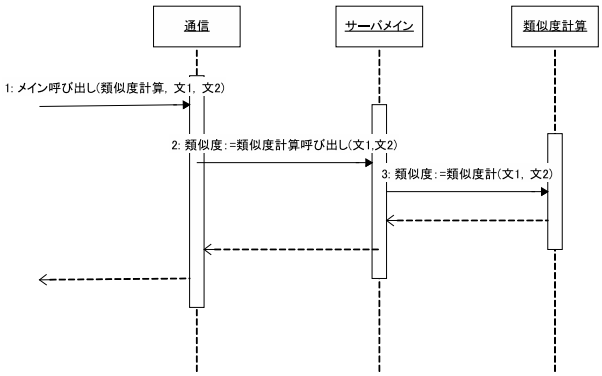

図 3.2: 類似度計算ツールのシーケンス図

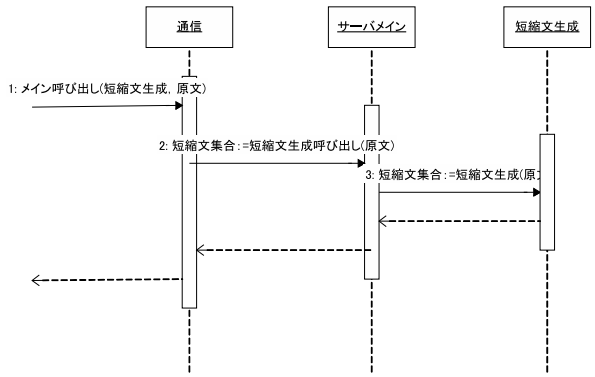

図 3.3: 文短縮ツールのシーケンス図

3.2.2 翻訳結果判断のシーケンス図

次に，機械翻訳が原文をうまく翻訳しているかの判断を行なう際の流れを説明する (図 3.4 参照).

ユーザから入力された原文をクライアント部が受け取り，外部翻訳サーバと通信をして折り返し翻訳を行なう．その結果として折り返し文が生成される．クライアント部は文解析サーバ内の類似度計算ツールを呼び出して，原文と折り返し文の間の類似度を 6 種類受け取る．類似度処理部でこれらの中からひとつを選び，クライアントメインに渡す．翻訳結果判断部がその類似度から原文がうまく翻訳されているかを判断し，結果がユーザに伝えられる．

3.2.3 機械翻訳不適箇所検出のシーケンス図

最後に，機械翻訳不適箇所を検出する際の流れをシーケンス図を用いて説明する (図 3.5 参照).

まず，翻訳に適していない原文がユーザインタフェースを通じてクライアン

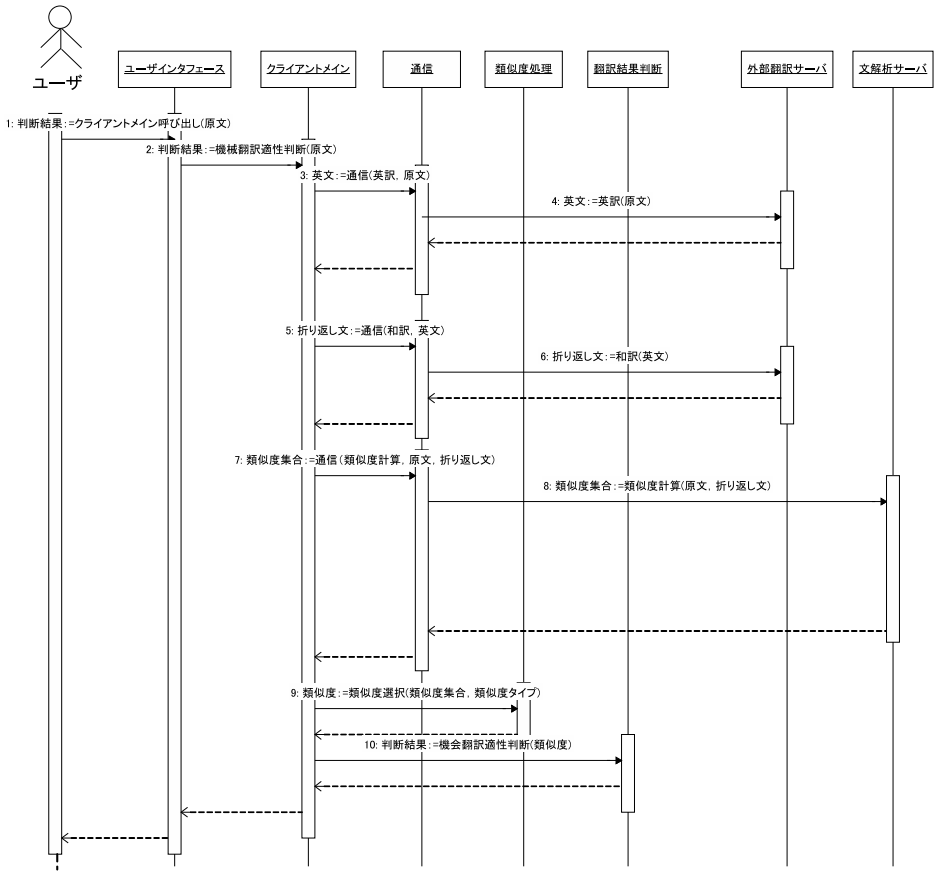

図 3.4: 翻訳結果判断のシーケンス図

トメインに渡される．クライアントメインは，原文の短縮文を生成するために，文解析サーバ内の文短縮ツールを呼び出し，原文の短縮文の集合を受け取る．クライアント部はこの短縮文の集合一つ一つに対して短-折類似度と原文類似度と 1st 原文類似度を計算する．これらの結果から，機械翻訳に適した短縮文の集合がクライアントメインに返ってくる．最後に，翻訳不適箇所検出部が機械翻訳に適した短縮文の集合の情報をもとに機械翻訳不適箇所を検出し，その結果をユーザに伝える．

3.3 SimTrans の各機能

次に，SimTrans の実際に動いている画面を用いながら各機能を説明する．

SimTrans では，ユーザに課題文を出し，ユーザはその課題文を機械翻訳に適したものになるようにリペアしていく．そのリペアを SimTrans が翻訳不適箇所の指摘でサポートするという形式をとる．

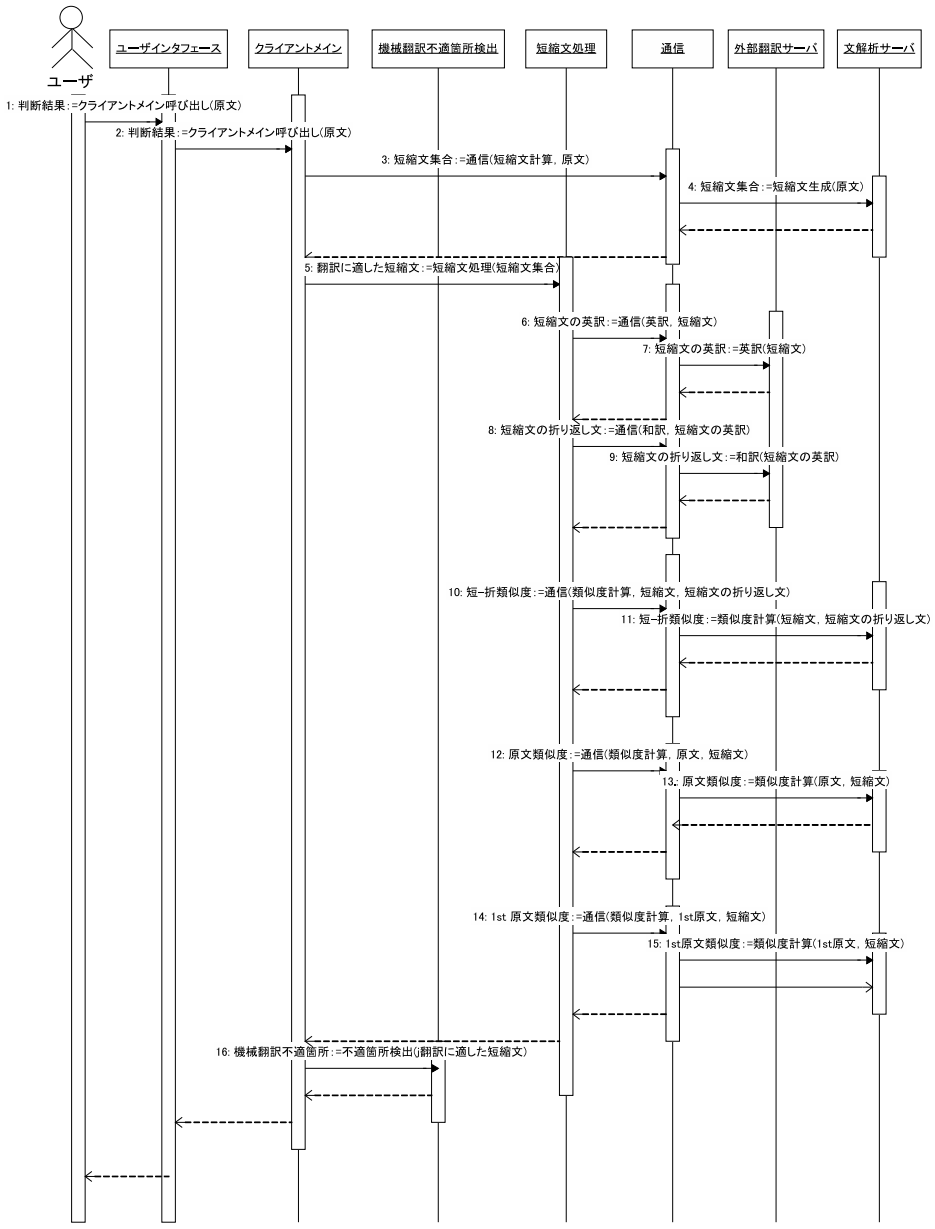

図 3.5: 機械翻訳不適箇所検出のシーケンス図

3.3.1 機械翻訳不適箇所の指摘

まず課題の文が図 3.6 の 1 に表示され，ユーザはその課題の文が機械翻訳によってうまく翻訳されるように課題文を修正して，原文を図 3.6 の 2 に入力する．原文を入力した上で図 3.6 の 3 にある翻訳ボタンを押すと，SimTrans は原文に対してコメントを三段階で図 3.6 の 5 に出力し，機械翻訳に不適な部分を図 3.6 の 4 のテキストボックスにハイライト表示してユーザに示す．もうひとつの結果として，図 3.6 の 6 には原文の英訳，折り返し文，原文と折り返し文との間

図 3.6: SimTrans のメイン画面

の類似度，SimTrans からユーザに出されたメッセージなどが表示される．ユーザは SimTrans に指摘された部分を参考に原文をリペアし，改めて SimTrans に入力として渡す．この操作をユーザが満足するまで繰り返す．

なお，機械翻訳に不適な箇所を判断する際の基準となる要素をユーザが変更して様々な実験を行えるようにした．変更できる要素は以下のような設定画面で設定する．

1. **クリアレベル**

   原文が翻訳に適しているかを判断するためのしきい値である．原文とその折り返し文との間のスコア (類似度を 100 点満点に置き換えたもの) がこのクリアレベルの値よりも大きければ,「うまく翻訳された」と判断し，そうでなければ「うまく翻訳されなかった」と判断する．

2. **文節重み付け**

   翻訳に不適な文節に与える重みに関する値で，この値が大きいほど翻訳に不適な文節とそうでない文節との間の差が大きくなる．

3. **短縮文しきい値**

   短縮文が機械翻訳に適しているかの判断をするための値である．短縮文とその折り返し文との間の類似度 (短-折類似度) がこの短縮文しきい値を超えていれば,「機械翻訳に適している」と判断し，そうでなければ「機械翻

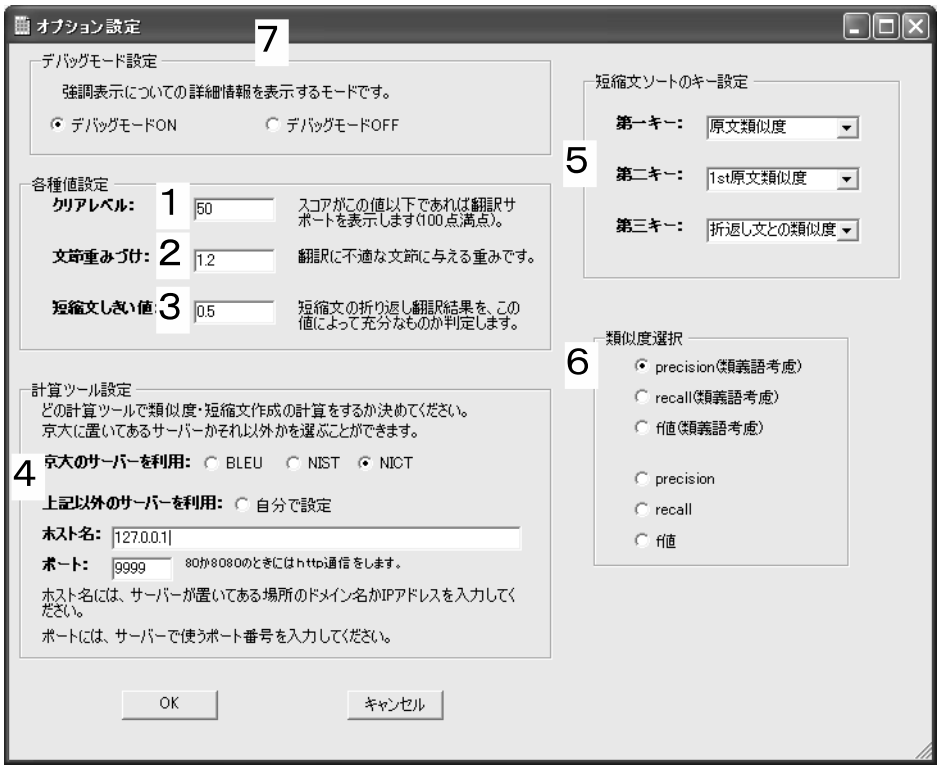

図 3.7: オプション設定画面

訳に適していない」と判断する．つまり，短縮文にとってのクリアレベル (上記) である．

4. **計算ツール設定**

   図 3.1 のとおり，SimTrans は文解析サーバと通信を行って機械翻訳を支援する．その際に，どこに置かれた文解析サーバと通信するのかを設定する部分である．初期値としては京大に設置したサーバになっているが，ユーザが自作したサーバやそれとは別の新たなサーバと通信するように設定可能である．IP アドレスとポート番号を指定することで通信するサーバを設定し，ポート番号が 80 か 8080 ならば HTTP 通信をし，それ以外であれば TCP 通信を行う．類似度計算ツールや文短縮ツールだけを変更してリペア支援を行う際に用いることを想定している．

5. **短縮文ソートのキー**

   2.2.3 節の中で「機械翻訳に適している短縮文」のソートについて，アルゴリズムを実装した際には三種類のキーの優先順位を自由に設定できるとしたが，その優先順位をここで設定する．ここでキーを設定することによっ

て，どのような短縮文を翻訳不適箇所の発見において重要視するのかが設定される．

6. **類似度計算の種類**
   図 3.1 の文解析サーバから SimTrans に送られてくる類似度の種類には，既に説明したとおり合計 6 つの類似度がある．その 6 つの類似度の中でどれを使うかを設定する．
7. **デバッグモード設定**
   次節で説明するデバッグモードを使うかどうかを設定する．

3.3.2 デバッグモード

通常のモードでは，SimTrans が具体的にどのようなデータを処理し，機械翻訳に不適な部分を検出したのかがわからないままに，機械翻訳に不適な部分の指摘だけを出力している．デバッグモードは SimTrans が扱ったデータを詳細に残し，SimTrans が機械翻訳に不適な箇所を検出するまでに行った処理過程を解析できるようにするモードである．

SimTrans がユーザから入力された原文の中で機械翻訳に不適な部分を検出し，ハイライト表示するという形式に変化はないが，(図 3.6 の 6) に表示される内容が変化する．通常のモードで出力されるデータに加え，以下の様なデータを残し，(図 3.6 の 6) のテキストボックスに出力する．

1. **原文類似ランキング**
   原文と短縮文ひとつひとつとの類似度によって短縮文をソートしたものである．通常のモードでは，原文と短縮文との間の類似度計算は,「短縮文しきい値」を超えた短縮文に対してのみ行われるが，すべての短縮文に対して計算し，その結果を出力する．
2. **1st 原文類似ランキング**
   1 回目の原文と短縮文ひとつひとつとの類似度によって短縮文をランキング付けしたものである．通常のモードでは，1 回目の原文と短縮文との間の類似度計算は，短縮文しきい値を超えた短縮文に対してのみ行われるが，すべての短縮文に対して計算し，その結果を出力する．
3. **短縮文翻訳適性ランキング**
   短縮文とその折り返し文との間の類似度によって短縮文をランキング付けしたもの．これによってしきい値を超える短縮文とそうでないものとを比較することができる．

4. **翻訳に適した短縮文リスト**
   短縮文とその折り返し文との間の類似度が「短縮文しきい値」を超えた短縮文のリストを，類似度順にソートして表示したものである．また，付加情報として，その短縮文に与えられた重み，原文には含まれているがその短縮文には含まれていない文節 (原文との差分)，原文との類似度，1 回目の原文との類似度，その短縮文の折り返し文との類似度も表示される．
5. **DIFF 得点ランキング**
   「翻訳に適した短縮文リスト」に要素である短縮文が持つ重みが，どの文節にどれだけ付与されるかを順に表示したものである．持っている情報としては上記の「翻訳に適した短縮文リスト」と変わらないが，データを見やすくするためにこれも表示している．
6. **文節不適ランキング**
   上記の「DIFF 得点ランキング」の結果が文節に反映されたものを，文節に与えられた重みでソートしたものである．SimTrans では，翻訳に不適な部分としてユーザーに指摘されるのは，最も大きな重みを与えられた文節だけであるが，それ以外の文節にどれだけ重みが与えられたかも調べることができる．

これらのデータを解析することで，SimTrans を含む図 3.1 の「機械翻訳支援システム」全体の解析を行うことができる．例えば，SimTrans が翻訳に不適な箇所を指摘できない際に，指摘できない原因を究明する必要が出てくる．しかし，SimTrans からの指摘箇所の情報と，原文の翻訳結果 (英文と折り返し文) だけで判断するのは，情報が少なく困難である．その際に上記 1 から 6 の情報を解析して原因の究明を行う．

他にも，本システムの中で重要な役割を果たしている，文短縮ツールと類似度計算ツールが SimTrans の要求に対してどのような出力を返したのかも解析することができるので，その結果からシステムの構成やツール間の入出力など，システム全体に関わる改良を行うことができる．

3.3.3 処理結果の XML による記録

すでに説明したとおり，SimTrans ではユーザの原文入力と SimTrans のリペア支援が交互に繰り返し行われる (図 2.1 参照)．そのため，SimTrans の処理過程のデータは複雑かつ大量なものになる．そのようなデータを解析しやすい形式で残しておくために，XML 形式でファイルに保存するようにした．

XML データの例として，例文:「明日，私は，会議に出席することになった.」に対して SimTrans を適用した例を図 3.8 に示す．それぞれのタグの意味については表 3.2 のとおりである．

図 3.8: XML データ例

表 3.2 の <goodTree> の説明にあるように，<goodTree> というデータは基本的には短縮文のデータであるが，類似度や与えられた重みなどの様々な情報を持つ．このようなオブジェクトデータをテキスト形式のような単純な形式で記録していくと単純に長くて解析しにくいデータとなる．しかし，XML を用いることでオブジェクトに対応したツリー構造で表現され，解析がしやすくなっている．また，SimTrans が扱うデータには <goodTree> のような，メンバを持ったオブジェクト型のデータが多数あり，その意味からも XML 形式のデータが有効である．

第 4 章　SimTrans の性能評価

この章では，SimTrans を実際に使用した結果について考察する．

今回の実験では 64 個の例文を用いたが，これらの例文は機械翻訳システムデザイナーが想定する日本語を利用者適応モデルで用いられるルールの例文から取り出したものである [4] ．機械翻訳を利用する過程は，利用者が自分の入力文に対するシステムの出力を解釈した結果，適切な出力文を得ることができるよ

表 3.2: XML データのタグの説明

タグ	説明
<name>	ユーザ名
<dateTime>	日付および時刻
<option>	オプション設定 (図 3.7 参照) の内容
<question>	課題文
<source>	ユーザが入力した原文
<mid>	機械翻訳によって目標言語に翻訳された原文
<target>	原文の折り返し文
<sim6>	原文とその折り返し文との間の類似度（6 種類あり）
<sprtmes>	SimTrans からユーザーに指摘されたメッセージ
<badparts>	SimTrans が翻訳に不適だと判断した文節
<bunsetuMatch>	原文を構成する文節ひとつひとつの ID 番号とその文節に与えられた重み
<orgRanking>	原文類似ランキング (3.3.2 参照)
<org_zeroRanking>	1st 原文類似ランキング (3.3.2 参照)
<fragRanking>	短縮文翻訳適性ランキング (3.3.2 参照)
<goodTree>	「短縮文しきい値」を超えた短縮文をソートしたデータ様々な類似度と短縮文に与えられた重みなどを持つ
<diffRanking>	DIFF 得点ランキング (3.3.2 参照)
<bunsetuRanking>	文節不適ランキング (3.3.2 参照)

うな入力文を作成するために有効なルールを獲得していく過程と考えられている，上記のルールとはこのルールである．

この 64 個の例文に対して SimTrans を適用した結果が付録の A.1 である．この結果を，以下の 4 つに分類し，その分類結果を表 4.1 に示す．

1. **理想的翻訳結果**

   例文とその折り返し文の類似度が 1 であり，かつ英訳もうまくいっている場合，機械翻訳不適箇所を指摘する必要はないので，このような例文は考察の対象外とした．

2. **適用例外**

   短縮文を生成して，短-折類似度を利用するという SimTrans の性質上，短縮文をうまく生成できないような例文には SimTrans は適用できない．そのため，主語述語等の文の構造を文短縮ツールによって解析できないような例文も考察の対象外とした．

3. **指摘可能**

   上記の 1 と 2 以外の例文の中で，指摘すべき箇所を SimTrans によって指摘

できるものである．

4. **指摘不可能**
   上記の 1 と 2 以外の例文の中で，指摘すべき箇所を SimTrans によって指摘できないものである．

表 4.1: 例文の分類結果

理想的翻訳結果	適用例外	指摘可能	指摘不可能	合計
5	6	31	22	64

この分類の中で，指摘可能に分類された例文の一部を付録より抜粋して表 4.2 に示す．

表 4.2: 指摘可能に分類された例文 (付録より抜粋)

番号	例文	指摘された箇所
	修正例	
評価	ルールの内容	
16	昨日、マトリックスリローデッドを 見た。	マトリックスリローデッドを
○	システムにとっての未知語の使用を避ける	
18	それは 漫画的発想 を持った人によって書かれた。	漫画的発想を
	それはユーモラスな発想を持った人によって書かれた。	
○	無理な造語の使用を避ける	

指摘可能に分類された例文の指摘箇所には，16 番の「マトリックスリローデッドを」や 18 番の「漫画的発想を」などのように，その文節単独で翻訳しにくいものが指摘されていることが多かった．表 4.2 には載っていないが，この他にも「テクニカルオーガナイザです。」や,「思わずそのまま」といった文節も指摘可能に分類された例文の中で指摘されていた．これらの文節はその他の文節との関係から翻訳しにくいのではなく，文節単独で翻訳されにくいので，これらの文節が含まれた短縮文は自然と翻訳しにくくなり，SimTrans がよく指摘できたと考えられる．

これ以外に，指摘可能に分類された例文の指摘箇所で多かったのは，文末である．SimTrans は 2.2.3 で述べたアルゴリズムで翻訳不適箇所が発見できない

際には，短-折類似度のしきい値を低くしてもう一度機械翻訳不適箇所を検出するが，これを実行した結果には文末を指摘することが多く，今回の例文の翻訳不適箇所が文末に多かったためであるとも考えられる．

次に，指摘不可能に分類された例文の一部を付録より抜粋して表 4.3 に示す．

表 4.3: 指摘不可能に分類された例文 (付録より抜粋)

番号	**例文**	**指摘された箇所**
	修正例	
評価	**ルールの内容**	
9	ああ、今日は寒い。	寒い。
	今日は寒い。	
×	感動詞の使用は避ける	
20	祖父の 米寿の お祝いをした。	した。
	祖父の八十八歳のお祝いをした。	
×	日本文化特有の事物名の使用を避ける	
41	私は自転車に乗りながら 携帯電話で話している少年を見た。	私は
	私は自転車に乗りながら、携帯電話で話している少年を見た。	
×	係り受け関係が明確になるような箇所に読点を挿入する	
44	買い物したり、観光したりして、楽しかった。	買い
×	名詞以外が並列した文章は避ける	
48	私は、その少年に財布を 盗まれた。	財布を
	その少年は、私の財布を盗んだ。	
×	受動態は避ける	

表 4.3 の内，9 番の例に関しては,「寒い」を英訳した「cold」を和訳すると「冷たい」になることが指摘不可能となった原因であったので，今回使用した機械翻訳と著者のアルゴリズムでは仕方のない結果であった．

20 番の例については,「米寿のお祝い」とその折り返し文である「八十八歳のお祝い」との間の類似度が 0 であり，うまく翻訳できていても，そうでないと判断された．このような例については類似度計算ツールの改良が進めば指摘できるようになると考えられる．

41 番の例については,「見た。」や「話している少年を見た。」という短縮文の短-折類似度が高いことに加えて,「自転車に乗りながら話している少年を見た。」という短縮文の短-折類似度も高い．このように，問題がある部分を含めて作られた短縮文が機械翻訳しやすい文になってしまうと，2.2.3 のアルゴリズムでは

指摘することは難しい.

44 番の例では,「買い物したり」という文節を「買い」と「物したり」という文節に区切っていて，その影響で不適箇所を指摘できなかった．つまり，短縮文生成ツールの出力に問題があった.

48 番の例についてはやや特殊で，SimTrans が指摘している部分が間違っているわけではないのである．SimTrans が指摘したのは「財布を」という文節であったが，今回の例文の指摘して欲しい箇所としては「盗まれた。」という部分である．しかし，この例文の英訳が「I was stolen in the purse by the boy.」であったことからも「財布を」という文節を指摘したことは間違いではない.

その他の指摘できなかった例文についても，上記の 5 つのいずれかのパターンに該当した．そのデータを表 4.4 に示す.

表 4.4: 例文の詳しい分類

指摘可能 (31)			**指摘不可能 (22)**				
難文節	文末	その他	英和	類似度	文短縮	アルゴリズム	指摘すべき場所
7	9	15	11	3	3	2	3

この結果から判断すると，機会翻訳や類似度計算ツールや文短縮ツールの性能が上がると今回は指摘できなかった例文に対しても SimTrans を用いて指摘できるようになることが予想される.

以上の結果全体から，文節単独で翻訳が難しいような文節や文末にある文節の機械翻訳不適箇所の指摘は SimTrans を用いるのに適していると言える．その一方で,「寒い」と「冷たい」の例のように折り返し結果がおかしくなってしまったり,「米寿お祝い」と「八十八歳のお祝い」の例のように類似度の値がおかしくなってしまうと，SimTrans は対応できない．しかし，この結果は類似度計算ツールや機械翻訳の性能が改良されることで改善されることが予想される．その他にも，文中に問題のある部分があった場合でも，その部分を含んだ短縮文の短-折類似度が高くなってしまうと，SimTrans では対応できない．このあたりの改良が SimTrans の今後の課題である.

第5章　おわりに

ICE2002の実験結果から，機械翻訳を解したコミュニケーションに関する以下のような問題点が見つかった．

- 機械翻訳がうまく翻訳していない際に，原文のどの部分が原因で機械翻訳がうまくいかないのかがわからない．

この問題に対して，自己主導型リペア支援システムを設計・実装することで問題点の解決に取り組んだ．

本研究はNICTと共同で行なわれ，NICTから提供していただいた類似度計算ツールと文短縮ツールの出力を利用することで，自己主導型リペア支援システムを実装したSimTransを開発した．このSimTransに対して64個の例文を適用し，その結果として以下のような成果が得られた．

1. SimTransに入力された文の翻訳不適箇所に,「マトリックスリローデッドを」や「漫画的発想を」のような，文節単独で翻訳しにくい文節が含まれているものは，機械翻訳不適箇所として指摘されやすい．
2. 文末の翻訳不適箇所は指摘されやすい．
3. 折り返し翻訳や類似度計算の値がおかしくなってしまうとそれに対応できない．
4. 文中に問題のある部分があっても，その部分を含んだ短縮文の短-折類似度が高くなってしまうと対応できない．

本研究で開発したSimTransは，類似度計算ツールや文短縮ツールの計算結果に大きく影響されるため，これらのツールの改良が進めば，今回は指摘不可能に分類された例文に対しても機械翻訳不適箇所を指摘できるようなることが予想される．これらのツールの改良はNICTが担当しているので，本研究の今後の課題としては，これらのツールの出力を処理する部分のアルゴリズムを改良することである．

謝辞

本研究について多大なるご指導，ご助言を賜りました石田亨教授に厚く御礼申し上げます．

また，共同研究でご指導，ご協力いただきました情報通信研究機構の井佐原均氏と内元清貴氏と竹内和広氏に深謝いたします．

そして，研究全般にわたり多くの有益なご助言をいただきました林田尚子氏と，技術面で多分にご協力いただきました中塚康介氏に心からお礼申し上げます.

最後になりますが, 日頃から多くの協力を頂きました石田研究室の皆様，誠にありがとうございました.

参考文献

[1] 野村早恵子, 石田亨, 船越要, 安岡美佳, 山下直美: アジアにおける異文化コラボレーション実験 2002:機械翻訳を介したソフトウェア開発, 情報処理, Vol. 44, No. 5, pp. 503–511 (2003).

[2] 内元清貴, 林田尚子, 石田亨, 井佐原均 (編): 機械翻訳可能性の自動評価, 言語処理学会第 11 回年次大会 (2005).

[3] 林田尚子, 石田亨 (編): 翻訳エージェントによる自己主導型リペア支援, JAWS2004 (2004).

[4] 山下直美, 坂本知子, 野村早恵子, 林田尚子, 石田亨, 井佐原均, 小倉健太郎, 林良彦, 石川開, 小谷克則, 島津美和子, 介弘達哉, 畠中伸敏, 富士秀, 船越要: 機械翻訳システムに対する話者適応の分析, 技術報告, 信学会 (2005). 投稿中.

付録

A.1 SimTrans の実行結果

4 章で説明した SimTrans に例文を適用した結果は A.1 のようになった.
なお，評価の記号の意味は以下のとおりである.
○：SimTrans のアルゴリズムで機械翻訳不適箇所を指摘できた例文
△：SimTrans の補助的なアルゴリズムで機械翻訳不適箇所を指摘できた例文
×：SimTrans では機械翻訳不適箇所を指摘できなかった例文
例外：完全に翻訳できてしまって問題のない文であるか，文の形が SimTrans の適用外であるために今回は対象外であるとした文のいずれかの例文

A.1: SimTrans に適用した例文

番号	**例文**	**指摘された箇所**
	修正例	
評価	**ルールの内容**	
1	ここで 話し合い を行う。	行う。
	ここで議論を行う。	
×	複数の意味を持つためうまく翻訳されない名詞を適切に置き換える	
2	彼女は落ち葉を 拾い集めた。	集めた。
	彼女は落ち葉を収集した。	
△	正しい翻訳ができない動詞を適切に置き換える	
3	他のサービスを 選んで ください。	※理想的翻訳
	他のサービスを選択してください。	
例外	動詞は名詞と解釈されないような形に書き換える	
4	その本を 読んでみた。	その
	その本を読んだ。	
×	補助動詞を伴うと上手く翻訳されない動詞を適切に置き換える	
5	これは大変珍しい 蝶である。	蝶である。
	これは大変珍しい蝶だ。	
△	形容動詞の置き換え	
6	休日で 人通りが少ない。	少ない。
	休日のため人通りが少ない。	
×	助動詞「で」の使用の回避	
7	是非、使って下さい。	是非、
	使って下さい。	
○	形容詞に書き換えられない副詞を削除する	

8	なぜなら 外は寒いから、コートを着た。	なぜなら
	コートを着た。なぜなら外は寒いからだ。	
○	文頭の接続詞は本来の述語句に係るようにする	
9	ああ、今日は寒い。	寒い。
	今日は寒い。	
×	感動詞の使用は避ける	
10	みなさん、こんにちは。	こんにちは。
	こんにちは。	
例外	呼び掛けの使用は避ける	
11	何を開発しようとしているか 報告してください。	報告してください。
	何を開発しようとしているかを報告してください。	
×	格助詞を明記する	
12	私は ケーキは 好きです。	好きです。
	私はケーキが好きです。	
×	主格でない副助詞（係助詞）「は」の使用は避ける	
13	オランダは チューリップが有名だ。	チューリップが
	オランダと言えばチューリップが有名です。	
×	話題提示の助詞「は」は、連用修飾句と同じ扱いにする (ただし 3 つ以上の述語句のある時は最後の述語句に係るものとする)	
14	傍若無人に 振る舞う。	振る舞う。
	勝手に振る舞う。	
×	熟語の使用を避ける	
15	雨が 降ったが、もう晴れている。	晴れている。
	雨が降ったけれども、もう晴れている。	
×	接続表現の書き換え	
16	昨日、マトリックスリローデッドを 見た。	マトリックスリローデッドを
○	システムにとっての未知語の使用を避ける	
17	私は テクニカルオーガナイザです。	テクニカルオーガナイザです。
	私は technical organizer です。	
○	正しい訳語が出ない外来語はアルファベット表示する	
18	それは 漫画的発想を 持った人によって書かれた。	漫画的発想を
	それはユーモラスな発想を持った人によって書かれた。	
○	無理な造語の使用を避ける	

19	今日は オペの 日だ。	今日は
	今日は手術の日だ。	
×	一般的な語句を用いる	
20	祖父の 米寿の お祝いをした。	した。
	祖父の八十八歳のお祝いをした。	
×	日本文化特有の事物名の使用を避ける	
21	鉛筆を つかって ください。	※理想的翻訳
	鉛筆を使って下さい。	
例外	同音異義語を持つ語句は漢字で書く	
22	ここで はきもの を脱いでください。	脱いでください。
	ここで履物を脱いでください。	
×	漢字表記可能な部分は漢字で書く	
23	私は ずっと、ここにいる必要がある。	ずっと、ここに
	私は長期間、ここにいる必要がある。	
○	多義語は一意に意味が特定できる表現に書き換える	
24	ソフトウェアを 起動の状態に せよ。	起動の
	ソフトウェアを起動せよ。	
○	削除してもその後の文の意味が変わらない語句は削除する	
25	ソフトウェアは、Windows、Photoshop、Lhaca などの ソフトウェアを含む。	Windows、
×	不必要な同一語句を削除する	
26	私の父は 大変ひどく 太っている。	大変ひどく
	私の父は大変太っている。	
○	不必要な同義語は削除する	
27	あの人のいう とうりです。	とうりです。
	あの人の言うとおりです。	
△	つづりが正しいことを確認する	
28	明日、買い物に 行きます。	買い物に
	明日、私は買い物に行きます。	
○	主語を明記する	
29	旧部品を 新部品に。	新部品に。
	旧部品を新部品に変更する。	
△	不完全な文の文末に述語を補完する	
30	日本語は英語に、英語は日本語に翻訳しなさい。	日本語は
	日本語は英語に翻訳し、英語は日本語に翻訳しなさい。	
×	主述関係を明確にするため、文の途中に述語を補完する	

31	車に 注意する。	※理想的翻訳
	車に注意を向ける。	
例外	動詞の書き換え	
32	病院は患者でいっぱいです。看護婦が 順番に 呼んでいます。	順番に
	病院は患者でいっぱいです。看護婦が患者を順番に呼んでいます。	
○	目的語を明記する	
33	大人しくて賢い、非常に珍しい 動物がいる。	大人しくて
	大人しくて賢い動物がいる。その動物は非常に珍しい。	
○	長い修飾語は適切な箇所で分割する	
34	入出力障害が発生した場合のオペレータの最初の作業は、障害媒体の復旧である。	入出力障害が
	入出力障害が発生した場合、最初にオペレータは障害媒体を復旧する。	
例外	連体修飾形を連用修飾形に変える	
35	非常にそのような ことは遺憾である。	非常にそのような
	そのようなことは非常に遺憾である。	
○	修飾語は直近の述語句、体言に係るようにする	
36	これが私は 正しいと思う。	これが
	私はこれが正しいと思う。	
○	修飾関係は非交差の条件を満たすものとする	
37	きれいな 桜が咲いた。	きれいな
	桜がきれいに咲いた。	
○	形容詞を副詞に変える	
38	プラントシステムが巨大になり、より高度な制御技術の導入が必要になった。	プラントシステムが より
	プラントシステムが巨大になった結果、より高度な制御技術の導入が必要になった。	
○	前節を、連体修飾語を含む副詞節に書き換える	
39	タイヤがスリップし、事故が起きた。	タイヤが
	タイヤがスリップしたために、事故が起きた。 事故が	
○	前節に形式名詞を挿入する	
40	再度今までの メッセージを消去します。	再度今までの
	再度、今までのメッセージを消去します。	
○	漢字が連なり一つの語として解釈されてしまうことを避けるため，語句の切れ目に読点を挿入する	

41	私は自転車に乗りながら 携帯電話で話している少年を見た。	私は
	私は自転車に乗りながら、携帯電話で話している少年を見た。	
×	係り受け関係が明確になるような箇所に読点を挿入する	
42	文字を印字し、スイッチを切る。	※理想的翻訳
	文字を印字する。そして、スイッチを切る。	
例外	連用中止を使用した文は，連用中止の前後で分割する	
43	次の 3 つを持参してください。1. 運動靴, 2. 帽子, 3. サングラス	持参してください。
	運動靴と帽子とサングラスを持参してください。	
例外	箇条書きを避ける	
44	買い物したり、観光したりして、楽しかった。	買い
×	名詞以外が並列した文章は避ける	
45	友達に、電話を、3 人した。	3 人
	3 人の友達に、電話をした。	
例外	文法的でない構造は避ける	
46	私は昨日映画を 鑑賞した。	鑑賞した。
	私は、昨日、映画を鑑賞した。	
×	単純な文法構造にする	
47	この写真は、その状況である。	その
	この写真は、その状況の証拠である。	
○	係りと受けを対応させる	
48	私は、その少年に財布を 盗まれた。	財布を
	その少年は、私の財布を盗んだ。	
×	受動態は避ける	
49	ネジを 締め付け 調整した。	調整した。
	ネジを締め付けてから、調整した。	
×	連用中止形を避け、適切な助詞を挿入する	
50	取り付け 完了。	指摘なし
	取り付けを完了した。	
例外	体言止めを使用しない	
51	新しい ものを 使っています。	※理想的翻訳
	新しいテレビを使っています。	
例外	具体的な表現に書き換える	
52	新しいのが 欲しい。	欲しい。
	新しいテレビが欲しい。	
×	準体助詞を具体的な表現に書き換える	

53	メモリ A に 表示されるメッセージを格納する。	格納する。
	表示されるメッセージをメモリ A に格納する。	
×	語順を変えて係り受けを明確にする	
54	タイトルは英語で書く ように して下さい。	書くようにして下さい。
	タイトルは英語で書いて下さい。	
△	形式名詞「もの」「こと」「よう」「ところ」を削除する	
55	思わずそのまま 投稿する。	思わずそのまま
	すぐ投稿する。	
○	複合的で、係り受け解析や適切な訳語の用意が困難な副詞句は、簡潔な表現に書き換える	
56	文章を明確にするのに 役立ちます。	役立ちます。
	文章を明確にします。	
△	文末複合表現の書き換え	
57	あなたの意見を 聞かせて下さい。	聞かせて下さい。
	あなたの意見を教えて下さい。	
△	使役＋依頼の複合様相表現の書き換え	
58	会いたかった のになぁ。	指摘なし
	会いたかった。でも会えなかった。	
例外	感情表現の使用を避ける	
59	両者には、雲泥の 差がある。	両者には、
	両者には、大きな差がある。	
×	慣用表現の使用を避ける	
60	彼は 超多忙な 毎日を送っている。	超多忙な
	彼は非常に多忙な毎日を送っている。	
○	スラング、方言の使用を避ける	
61	長い目で見る。	長い
	長期的に見る。	
○	比喩の使用を避ける	
62	宜しくお願いします。	お願いします。
例外	独特の言い回しを避ける	

63	その意義が 疑われかねません。	疑われかねません。
	その意義が疑われます。	
△	複雑な否定表現を避ける	
64	あなたに感謝 申し上げます。	申し上げます。
	あなたに感謝します。	
△	敬語表現は避ける	

A.2 ソースコード

A.2.1 SimtransForm.cs

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.IO;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading;

namespace SimTrans
{
/// <summary>
/// Form1 の概要の説明です。
/// </summary>
public class SimtransForm : System.Windows.Forms.Form
{
private System.Windows.Forms.Label label1;
private System.Windows.Forms.Label label2;
private System.Windows.Forms.Button clearButton;
private System.Windows.Forms.Button submitButton;
private System.Windows.Forms.Label label4;
private System.Windows.Forms.Label label5;
private System.ComponentModel.IContainer components;

//変数

LogMain log; // Log.cs 内
LogData crntlogdata;
ToolFunction tool = new ToolFunction();
Translation translation; // Translation.cs 内
Similar similar; // Similar.cs 内
CompresSentence compressnt; // SompresSentence.cs 内
SimCmprs simcmprs; // SimCmprs.cs 内　類似度＋日本語のまずさスコア
private System.Windows.Forms.Timer writelogTimer;
private System.Windows.Forms.Button testtransButton;
private System.Windows.Forms.RichTextBox logRichTextBox;
String username;
String simmethod = "BLEU"; //類似度計算の計算手法 BLEU or NIST or CS or NICT
String hostName = "";//デフォルトは空にしておく。
int hostPort = 80;
bool debugMode = false;
private ArrayList GoodTreeList = new ArrayList();//一定ポイント以上の類似度を持つ短縮文を記録しておくためのログ。
private ArrayList fragList = new ArrayList();//
private ArrayList org_try_list = new ArrayList();//原文-短縮文の類似度情報を保存
private ArrayList org_zero_list = new ArrayList();//1st 原文-短縮文の類似度情報を保存
string first_key = "self_sim";//この三つで GTD がソートされるキーの順番が決まる。
string second_key = "org_sim";
string third_key = "try_sim";
LastWord lastword = new LastWord();
int transflg = 0;
Thread threadPointer;
String org_zero = "";
double weightpoint = 0;
double clrsimpoint = 0.1; // 類似度が十分に高いかどうかの基準値
double fragclrpoint = 0.0; //短縮文を評価するのに用いる閾値。
int currentNumber;// 翻訳課題番号
string simtype = "recall";
private System.Windows.Forms.RichTextBox kadaiRichTextBox;
private System.Windows.Forms.RichTextBox sourceRichTextBox;
private System.Windows.Forms.RichTextBox checkmesRichTextBox;
private System.Windows.Forms.ContextMenu cpcutpastesourceContextMenu;
private System.Windows.Forms.MenuItem cutsourceMenuItem;
private System.Windows.Forms.MenuItem pastesourceMenuItem;
private System.Windows.Forms.MenuItem cpsourceMenuItem;
private System.Windows.Forms.ContextMenu cpkadaiContextMenu;
private System.Windows.Forms.MenuItem cpkadaiMenuItem;
```

```
private System.Windows.Forms.ContextMenu cpcheckmesContextMenu;
private System.Windows.Forms.MenuItem cpcheckmesMenuItem;
private System.Windows.Forms.ContextMenu cplogContextMenu;
private System.Windows.Forms.MenuItem cplogMenuItem;
private System.Windows.Forms.Button configButton;

double [] point;// 分子
double [] appear;// 分母
double [] APfrac;// 分子/分母

int repaircount; // リペア回数
string badparts;
int badparts_length;
bns [] bunsetu_match;
string [] bunsetu_tree;
private System.Windows.Forms.Label label6;
private System.Windows.Forms.GroupBox groupBox1;
private System.Windows.Forms.Label label3;
private System.Windows.Forms.Button saveButton;
private HighlightRichTextBox.HighlightRichTextBox highlight;
private System.Windows.Forms.Button transCancel;


public SimtransForm()
{
InitializeComponent();

translation = new Translation();
similar = new Similar();
compressnt = new CompresSentence();
simcmprs = new SimCmprs();
repaircount = 1;
badparts = "";
badparts_length = 0;

Icon myIcon = new Icon(GetType(), "App.ico");
}

protected override void Dispose( bool disposing )
{
if( disposing )
{
if (components != null)
{
components.Dispose();
}
}
base.Dispose( disposing );
}

#region Windows フォーム デザイナで生成されたコード
/// <summary>
/// デザイナ サポートに必要なメソッドです。このメソッドの内容を
/// コード エディタで変更しないでください。
/// </summary>
private void InitializeComponent()
{
this.components = new System.ComponentModel.Container();
System.Resources.ResourceManager resources = new System.Resources.ResourceManager(typeof(SimtransForm));
this.label1 = new System.Windows.Forms.Label();
this.label2 = new System.Windows.Forms.Label();
this.clearButton = new System.Windows.Forms.Button();
this.submitButton = new System.Windows.Forms.Button();
this.label4 = new System.Windows.Forms.Label();
this.label5 = new System.Windows.Forms.Label();
this.writelogTimer = new System.Windows.Forms.Timer(this.components);
this.testtransButton = new System.Windows.Forms.Button();
this.logRichTextBox = new System.Windows.Forms.RichTextBox();
this.cplogContextMenu = new System.Windows.Forms.ContextMenu();
this.cplogMenuItem = new System.Windows.Forms.MenuItem();
this.kadaiRichTextBox = new System.Windows.Forms.RichTextBox();
```

```
this.cpkadaiContextMenu = new System.Windows.Forms.ContextMenu();
this.cpkadaiMenuItem = new System.Windows.Forms.MenuItem();
this.sourceRichTextBox = new System.Windows.Forms.RichTextBox();
this.cpcutpastesourceContextMenu = new System.Windows.Forms.ContextMenu();
this.cpsourceMenuItem = new System.Windows.Forms.MenuItem();
this.cutsourceMenuItem = new System.Windows.Forms.MenuItem();
this.pastesourceMenuItem = new System.Windows.Forms.MenuItem();
this.checkmesRichTextBox = new System.Windows.Forms.RichTextBox();
this.cpcheckmesContextMenu = new System.Windows.Forms.ContextMenu();
this.cpcheckmesMenuItem = new System.Windows.Forms.MenuItem();
this.configButton = new System.Windows.Forms.Button();
this.label6 = new System.Windows.Forms.Label();
this.groupBox1 = new System.Windows.Forms.GroupBox();
this.highlight = new HighlightRichTextBox.HighlightRichTextBox();
this.label3 = new System.Windows.Forms.Label();
this.saveButton = new System.Windows.Forms.Button();
this.transCancel = new System.Windows.Forms.Button();
this.groupBox1.SuspendLayout();
this.SuspendLayout();
//
// label1
//
this.label1.Location = new System.Drawing.Point(8, 24);
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(40, 23);
this.label1.TabIndex = 8;
this.label1.Text = "課題";
this.label1.Click += new System.EventHandler(this.label1_Click);
//
// label2
//
this.label2.Location = new System.Drawing.Point(8, 104);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(32, 16);
this.label2.TabIndex = 9;
this.label2.Text = "原文";
//
// clearButton
//
this.clearButton.Location = new System.Drawing.Point(8, 136);
this.clearButton.Name = "clearButton";
this.clearButton.Size = new System.Drawing.Size(56, 32);
this.clearButton.TabIndex = 8;
this.clearButton.Text = "課題文に戻す";
this.clearButton.Click += new System.EventHandler(this.clearButton_Click);
//
// submitButton
//
this.submitButton.Location = new System.Drawing.Point(400, 352);
this.submitButton.Name = "submitButton";
this.submitButton.Size = new System.Drawing.Size(64, 24);
this.submitButton.TabIndex = 6;
this.submitButton.Text = "次の課題";
this.submitButton.Click += new System.EventHandler(this.submitButton_Click);
//
// label4
//
this.label4.Font = new System.Drawing.Font("MS UI Gothic", 9F, System.Drawing.FontStyle.
Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(128)));
this.label4.Location = new System.Drawing.Point(8, 24);
this.label4.Name = "label4";
this.label4.Size = new System.Drawing.Size(40, 16);
this.label4.TabIndex = 1;
this.label4.Text = "翻訳";
//
// label5
//
this.label5.Location = new System.Drawing.Point(8, 184);
this.label5.Name = "label5";
this.label5.Size = new System.Drawing.Size(56, 16);
this.label5.TabIndex = 1;
```

```
this.label5.Text = "メッセージ";
//
// writelogTimer
//
this.writelogTimer.Interval = 180000;
this.writelogTimer.Tick += new System.EventHandler(this.writelogTimer_Tick);
//
// testtransButton
//
this.testtransButton.Location = new System.Drawing.Point(496, 112);
this.testtransButton.Name = "testtransButton";
this.testtransButton.Size = new System.Drawing.Size(64, 24);
this.testtransButton.TabIndex = 3;
this.testtransButton.Text = "翻訳";
this.testtransButton.Click += new System.EventHandler(this.testtransButton_Click);
//
// logRichTextBox
//
this.logRichTextBox.BackColor = System.Drawing.SystemColors.Control;
this.logRichTextBox.ContextMenu = this.cplogContextMenu;
this.logRichTextBox.Location = new System.Drawing.Point(568, 32);
this.logRichTextBox.Name = "logRichTextBox";
this.logRichTextBox.ReadOnly = true;
this.logRichTextBox.Size = new System.Drawing.Size(376, 512);
this.logRichTextBox.TabIndex = 12;
this.logRichTextBox.TabStop = false;
this.logRichTextBox.Text = "試行の経過がここに表示されます．";
//
// cplogContextMenu
//
this.cplogContextMenu.MenuItems.AddRange(new System.Windows.Forms.MenuItem[] {
 this.cplogMenuItem});
//
// cplogMenuItem
//
this.cplogMenuItem.Index = 0;
this.cplogMenuItem.Shortcut = System.Windows.Forms.Shortcut.CtrlP;
this.cplogMenuItem.Text = "コピー";
this.cplogMenuItem.Click += new System.EventHandler(this.cplogMenuItem_Click);
//
// kadaiRichTextBox
//
this.kadaiRichTextBox.BackColor = System.Drawing.SystemColors.Info;
this.kadaiRichTextBox.ContextMenu = this.cpkadaiContextMenu;
this.kadaiRichTextBox.Location = new System.Drawing.Point(80, 24);
this.kadaiRichTextBox.Name = "kadaiRichTextBox";
this.kadaiRichTextBox.ReadOnly = true;
this.kadaiRichTextBox.Size = new System.Drawing.Size(400, 64);
this.kadaiRichTextBox.TabIndex = 1;
this.kadaiRichTextBox.TabStop = false;
this.kadaiRichTextBox.Text = "課題";
this.kadaiRichTextBox.TextChanged += new System.EventHandler(this.kadaiRichTextBox_TextChanged);
//
// cpkadaiContextMenu
//
this.cpkadaiContextMenu.MenuItems.AddRange(new System.Windows.Forms.MenuItem[] {
   this.cpkadaiMenuItem});
//
// cpkadaiMenuItem
//
this.cpkadaiMenuItem.Index = 0;
this.cpkadaiMenuItem.Shortcut = System.Windows.Forms.Shortcut.CtrlC;
this.cpkadaiMenuItem.Text = "コピー";
this.cpkadaiMenuItem.Click += new System.EventHandler(this.cpkadaiMenuItem_Click);
//
// sourceRichTextBox
//
this.sourceRichTextBox.BackColor = System.Drawing.SystemColors.Window;
this.sourceRichTextBox.ContextMenu = this.cpcutpastesourceContextMenu;
this.sourceRichTextBox.ImeMode = System.Windows.Forms.ImeMode.On;
this.sourceRichTextBox.Location = new System.Drawing.Point(80, 104);
```

```
this.sourceRichTextBox.Name = "sourceRichTextBox";
this.sourceRichTextBox.Size = new System.Drawing.Size(400, 72);
this.sourceRichTextBox.TabIndex = 2;
this.sourceRichTextBox.Text = "入力した文章";
//
// cpcutpastesourceContextMenu
//
this.cpcutpastesourceContextMenu.MenuItems.AddRange(new System.Windows.Forms.MenuItem[] {
this.cpsourceMenuItem,
this.cutsourceMenuItem,
this.pastesourceMenuItem});
//
// cpsourceMenuItem
//
this.cpsourceMenuItem.Index = 0;
this.cpsourceMenuItem.Shortcut = System.Windows.Forms.Shortcut.CtrlC;
this.cpsourceMenuItem.Text = "コピー";
this.cpsourceMenuItem.Click += new System.EventHandler(this.cpsourceMenuItem_Click);
//
// cutsourceMenuItem
//
this.cutsourceMenuItem.Index = 1;
this.cutsourceMenuItem.Shortcut = System.Windows.Forms.Shortcut.CtrlX;
this.cutsourceMenuItem.Text = "切り取り";
this.cutsourceMenuItem.Click += new System.EventHandler(this.cutsourceMenuItem_Click);
//
// pastesourceMenuItem
//
this.pastesourceMenuItem.Index = 2;
this.pastesourceMenuItem.Shortcut = System.Windows.Forms.Shortcut.CtrlP;
this.pastesourceMenuItem.Text = "貼り付け";
this.pastesourceMenuItem.Click += new System.EventHandler(this.pastesourceMenuItem_Click);
//
// checkmesRichTextBox
//
this.checkmesRichTextBox.BackColor = System.Drawing.SystemColors.Info;
this.checkmesRichTextBox.ContextMenu = this.cpcheckmesContextMenu;
this.checkmesRichTextBox.Location = new System.Drawing.Point(72, 368);
this.checkmesRichTextBox.Name = "checkmesRichTextBox";
this.checkmesRichTextBox.ReadOnly = true;
this.checkmesRichTextBox.Size = new System.Drawing.Size(448, 152);
this.checkmesRichTextBox.TabIndex = 9;
this.checkmesRichTextBox.TabStop = false;
this.checkmesRichTextBox.Text = "ここには，より良い翻訳を行う為のメッセージが表示されます．";
this.checkmesRichTextBox.TextChanged += new System.EventHandler(this.checkmesRichTextBox_TextChanged);
//
// cpcheckmesContextMenu
//
this.cpcheckmesContextMenu.MenuItems.AddRange(new System.Windows.Forms.MenuItem[] {
  this.cpcheckmesMenuItem});
//
// cpcheckmesMenuItem
//
this.cpcheckmesMenuItem.Index = 0;
this.cpcheckmesMenuItem.Shortcut = System.Windows.Forms.Shortcut.CtrlP;
this.cpcheckmesMenuItem.Text = "コピー";
this.cpcheckmesMenuItem.Click += new System.EventHandler(this.cpcheckmesMenuItem_Click);
//
// configButton
//
this.configButton.Location = new System.Drawing.Point(568, 8);
this.configButton.Name = "configButton";
this.configButton.Size = new System.Drawing.Size(48, 24);
this.configButton.TabIndex = 5;
this.configButton.Text = "設定";
this.configButton.Click += new System.EventHandler(this.configButton_Click);
//
// label6
//
this.label6.Font = new System.Drawing.Font("MS UI Gothic", 9F, System.Drawing.FontStyle.
Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(128)));
```

```
this.label6.Location = new System.Drawing.Point(8, 40);
this.label6.Name = "label6";
this.label6.Size = new System.Drawing.Size(48, 16);
this.label6.TabIndex = 1;
this.label6.Text = "サポート";
//
// groupBox1
//
this.groupBox1.Controls.Add(this.highlight);
this.groupBox1.Controls.Add(this.label5);
this.groupBox1.Controls.Add(this.label4);
this.groupBox1.Controls.Add(this.label6);
this.groupBox1.Controls.Add(this.label3);
this.groupBox1.Controls.Add(this.submitButton);
this.groupBox1.Location = new System.Drawing.Point(8, 184);
this.groupBox1.Name = "groupBox1";
this.groupBox1.Size = new System.Drawing.Size(536, 392);
this.groupBox1.TabIndex = 14;
this.groupBox1.TabStop = false;
//
// highlight
//
this.highlight.BackColor = System.Drawing.SystemColors.Info;
this.highlight.Location = new System.Drawing.Point(64, 24);
this.highlight.Name = "highlight";
this.highlight.ReadOnly = true;
this.highlight.Size = new System.Drawing.Size(448, 128);
this.highlight.TabIndex = 8;
this.highlight.TabStop = false;
this.highlight.Text = "ここには，原文の修正個所候補を提示します．";
//
// label3
//
this.label3.Location = new System.Drawing.Point(72, 352);
this.label3.Name = "label3";
this.label3.Size = new System.Drawing.Size(224, 16);
this.label3.TabIndex = 0;
this.label3.Text = "翻訳に満足したら、次の課題を選択ください。";
//
// saveButton
//
this.saveButton.Location = new System.Drawing.Point(560, 552);
this.saveButton.Name = "saveButton";
this.saveButton.Size = new System.Drawing.Size(40, 24);
this.saveButton.TabIndex = 7;
this.saveButton.Text = "保存";
this.saveButton.Click += new System.EventHandler(this.saveButton_Click);
//
// transCancel
//
this.transCancel.Location = new System.Drawing.Point(496, 152);
this.transCancel.Name = "transCancel";
this.transCancel.Size = new System.Drawing.Size(64, 23);
this.transCancel.TabIndex = 4;
this.transCancel.Text = "キャンセル";
this.transCancel.Click += new System.EventHandler(this.transCancel_Click);
//
// SimtransForm
//
this.AutoScaleBaseSize = new System.Drawing.Size(5, 12);
this.BackColor = System.Drawing.SystemColors.Control;
this.ClientSize = new System.Drawing.Size(968, 590);
this.Controls.Add(this.transCancel);
this.Controls.Add(this.configButton);
this.Controls.Add(this.checkmesRichTextBox);
this.Controls.Add(this.sourceRichTextBox);
this.Controls.Add(this.kadaiRichTextBox);
this.Controls.Add(this.logRichTextBox);
this.Controls.Add(this.clearButton);
this.Controls.Add(this.label1);
this.Controls.Add(this.label2);
```

```
this.Controls.Add(this.testtransButton);
this.Controls.Add(this.groupBox1);
this.Controls.Add(this.saveButton);
this.Icon = ((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
this.Name = "SimtransForm";
this.Text = "日英翻訳の翻訳前サポート";
this.Load += new System.EventHandler(this.SimtransForm_Load);
this.Closed += new System.EventHandler(this.SimtransForm_Closed);
this.groupBox1.ResumeLayout(false);
this.ResumeLayout(false);

}
#endregion

[STAThread]
static void Main()
{
Application.Run(new SimtransForm());
}

private void SimtransForm_Load(object sender, System.EventArgs e)
{
InitUser();
if(!File.Exists(username + ".xml"))
{
log = new LogMain(username);
Questions.UserQuestion(log);

currentNumber = 0;
if(log != null)
{
log.logResultList.Add(new LogResult(currentNumber));
}
}
else
{
log = LogTool.ReadLog(username + ".xml");
if(log.postQuestionList.Count > 0)
{
                    MessageBox.Show("すでに同じ名前の人がいます");
this.Close();
}
else if(log.logResultList.Count > 0)
{
currentNumber = log.logResultList.Count - 1;
if(log.logResultList[currentNumber].logDataList.Count > 0)
{
int tail = log.logResultList[currentNumber].logDataList.Count - 1;
this.sourceRichTextBox.Text = log.logResultList[currentNumber].logDataList[tail].source;
}
}
else
{
currentNumber = 0;
}
}
LoadSampleText(currentNumber);


////////////////////////////設定の初期値読み込み//////////////////////////
if (File.Exists("_cal_option.txt"))
{
simmethod = Get_caloption("SimMethod");
clrsimpoint = Double.Parse(Get_caloption("ClrsimPoint"));
clrsimpoint = clrsimpoint / 100;
weightpoint = Double.Parse(Get_caloption("WeightPoint"));

fragclrpoint = Double.Parse(Get_caloption("FragClrPoint"));

debugMode = Boolean.Parse(Get_caloption("DebugMode"));
hostName = Get_caloption("hostName");
```

```
hostPort = (int)(Double.Parse(Get_caloption("hostPort")));

first_key = Get_caloption("first");
second_key = Get_caloption("second");
third_key = Get_caloption("third");

simtype = Get_caloption("simType");
}
//////////////////////////////初期値読み込み完了//////////////////////////////
this.highlight.Text = "ここには，原文の修正個所候補を提示します．";
this.writelogTimer.Start();
}

private string Get_caloption(string optionname)
{
TextReader reader = null;
string options = "";

try
{
reader = new StreamReader("_cal_option.txt", System.Text.Encoding.GetEncoding(932)); //932 SJIS
options = reader.ReadToEnd();
}
catch(FileNotFoundException)
{
return null;
}
finally
{
if(reader != null)
{
reader.Close();
}
}

string [] optionlist = Regex.Split(options, "\r\n");
foreach ( string option in optionlist)
{
string [] optionvalue = Regex.Split(option, ",");
if (optionname == optionvalue[0]) return optionvalue[1];
}
return null;
}

private void InitUser()
{
SetNameForm setName = new SetNameForm();

switch(setName.ShowDialog())
{
case DialogResult.OK:
username = setName.usernameTextBox.Text;
this.Text = "SimTrans - " + username + "さん";
break;
case DialogResult.Cancel:
this.Text = "お試し中";
log = null;
break;
}
}



private void SetOption() // 設定の変更
{
SetoptionForm setOption =
new SetoptionForm(simmethod, clrsimpoint, weightpoint, fragclrpoint,hostName,hostPort,debugMode,
first_key, second_key, third_key,simtype);

switch (setOption.ShowDialog())//SetoptionForm ウィンドウの結果をもらう
{
```

```
case DialogResult.OK:

clrsimpoint = (setOption.Setclrsimpoint())/100;
weightpoint = setOption.Setweight();
fragclrpoint = setOption.SetFragClrPoint();

simmethod = setOption.Setcalmethod();//BLEU, NIST, CS, NICT, SELF
hostName = setOption.SetHostName();//host 情報の獲得
hostPort = setOption.SetHostPort();
                    debugMode = setOption.setMode();

first_key = setOption.setFirstKey();
second_key = setOption.setSecondKey();
                    third_key = setOption.setThirdKey();

if(simmethod == "BLEU"  || simmethod == "NIST")simtype = "recall";
//値が一つしか出ないときは recallってことにしておく。
else simtype = setOption.setSimtype();
break;
case DialogResult.Cancel:
break;
}
///////以下設定されたオプションに不備があるときのメッセージ
if(simmethod.Equals("CS"))
{
System.Windows.Forms.MessageBox.Show("CS 研の類似度計算はただいま使えません");
SetOption();
}
if(simmethod.Equals("SELF") && hostName.Equals(""))
{
System.Windows.Forms.MessageBox.Show("「自分で設定」を選ばれた方は「ホスト」も指定してください");
SetOption();
}
if((first_key==second_key) || (second_key==third_key) || (third_key==first_key)){
System.Windows.Forms.MessageBox.Show("ソートのキーは重ならないようにしてください。");
SetOption();
}
}


private bool LoadSampleText(int n)
{
TextReader reader = null;

try
{
reader = new StreamReader("_sample"+n.ToString("00")+".txt", System.Text.Encoding.GetEncoding(932));
this.sourceRichTextBox.Text = this.kadaiRichTextBox.Text = reader.ReadToEnd();
this.Text = (n+1).ToString() + "問目/" + Directory.GetFiles("./", "_sample??.txt").Length.ToString() + "問中";
return true;
}
catch(FileNotFoundException)
{
return false;
}
finally
{
if(reader != null)
{
reader.Close();
}
}
}



private void writelogTimer_Tick(object sender, System.EventArgs e)
{
Thread thread = new Thread(new ThreadStart(WriteLogThread));
thread.Start();
}
```

```
private void WriteLogThread()
{
LogTool.WriteLog(log, username + ".xml");
}

private void clearButton_Click(object sender, System.EventArgs e)
{
this.sourceRichTextBox.Text = this.kadaiRichTextBox.Text;
}


private void saveButton_Click(object sender, System.EventArgs e)
{
LogTool.WriteLog(log, username + ".xml");//xml ファイルにも書き込み。

            string current_directory = Directory.GetCurrentDirectory();

SaveFileDialog dialog=new SaveFileDialog();
dialog.InitialDirectory = current_directory;//保存先のデフォルトをカレントに設定。
dialog.FileName="新しいファイル.txt";
if(dialog.ShowDialog() == DialogResult.OK)
{
System.IO.Stream stream;
stream=dialog.OpenFile();
if(stream!=null)
{
System.IO.StreamWriter sw=new System.IO.StreamWriter(stream);
sw.Write(this.logRichTextBox.Text.Replace("\n","\r\n"));
sw.Close();
stream.Close();
}
}
Directory.SetCurrentDirectory(current_directory);
}


private void testtransButton_Click(object sender, System.EventArgs e)
{

if(transflg == 0)
{
transflg = 1;
this.Cursor = System.Windows.Forms.Cursors.WaitCursor;
threadPointer = new Thread(new ThreadStart(testtrans));
                threadPointer.Start();
}
return;
}

private void testtrans()
        {
string mid,  target;
Similar sim6 = new Similar();

this.Cursor = System.Windows.Forms.Cursors.WaitCursor;

//初期化
badparts = "";
badparts_length = 0;
this.checkmesRichTextBox.Text = "ここには，より良い翻訳を行う為のメッセージが表示されます．";
this.highlight.Clear();
this.highlight.Text = "ここには，原文の修正個所候補を提示します．";//翻訳サポートにメッセージを表示

            org_try_list.Clear();
org_zero_list.Clear();
fragList.Clear();

this.point = new double [] {};
this.appear = new double [] {};
this.APfrac = new double [] {};
```

```
//初期化ここまで

if (this.sourceRichTextBox.Text.Equals(""))
{
MessageBox.Show("文章を入力して下さい。");
}
else
{
if(org_zero.Equals(""))//一回目に入力した文を記憶しておく（二回目以降の原文とは位置づけが異なるため）
{
org_zero = Regex.Replace(this.sourceRichTextBox.Text, " |  |\r\n", "");//半角スペース，全角スペース，改行の消去.
}

mid = translation.Translate(this.sourceRichTextBox.Text, "J", "E");//mid:和→英を実行。あとで英→和もする。
mid = Regex.Replace(mid, "\\**", "");//翻訳結果にアスタリスクが入ってくる場合があるからその除去を行ってる。

if (mid.Equals(""))
{
MessageBox.Show("翻訳できませんでした。文章を変更して下さい。");
}
else
{
target = translation.Translate(mid, "E", "J");//折り返し翻訳

mid = Regex.Replace(mid, "\r\n|\n", "");

target = Regex.Replace(target, " |  |\r\n", "");//半角スペース，全角スペース，改行の消去.

string source = Regex.Replace(this.sourceRichTextBox.Text, " |  |\r\n|\n", "");
this.sourceRichTextBox.Text = source;//原文の整形

// CS 研のツールは半角が使えないので，全角に変換する必要がある．ツールが CS 研以外であれば考慮しない.
if (simmethod.Equals("CS"))
{
source = Trans_hankaku_zenkaku(source);//source の全角化
target = Trans_hankaku_zenkaku(target);//target の全角化
}

if (source.Equals("") || target.Equals(""))
{
sim6.precision = 0;
sim6.recall = 0;
sim6.f_value = 0;
sim6.precision_zero = 0;
sim6.recall_zero = 0;
sim6.f_value_zero = 0;
}
else
{
string sim = similar.Cal_similar(source, target, simmethod, hostName, hostPort);
                        sim6 = sim6.setSimPoint(sim);

if(simmethod.Equals("NIST"))//NIST のときには類似度が 10 倍で返ってくるから直す必要あり。
{
sim6.recall = sim6.recall / 10;
}
}
//ここまでで原文とその折り返し翻訳との間の類似度が計算されている。


//////////////////////原文を構成する文節の獲得////////////////////////////////
string bunsetu_line = compressnt.Get_bundanpen(source, "bunsetu",simmethod,hostName,hostPort);
string [] IDsource = new string [2];
IDsource = Regex.Split(bunsetu_line,"---tansyukubun start---\n");
string IDbunsetu = IDsource[0];//(ID 文節) のセットが\n 区切りで並んでる
string IDtemp = IDsource[1];
IDtemp = Regex.Replace(IDtemp,"\n---tansyukubun end---\n*","");

bunsetu_match = new bns [Regex.Split(IDbunsetu, "\n").Length-1];
bunsetu_match = bns.get_bunsetu_match(IDbunsetu);//ID と文節の対応表の完成
```

```
bunsetu_tree = new string [Regex.Split(IDtemp,"\n").Length-1];
bunsetu_tree = Regex.Split(IDtemp,"\n");
//////////////////////原文の文節の獲得完了（bunsetu_match [] に入った）//////

this.point = new double [bunsetu_match.Length];
this.appear = new double [bunsetu_match.Length];
this.APfrac = new double [bunsetu_match.Length];


if(log != null)
{
string question = kadaiRichTextBox.Text;

crntlogdata = new LogData(sim6, question, source, mid, target,
clrsimpoint, weightpoint, fragclrpoint, simmethod, hostName, hostPort,
first_key, second_key, third_key, simtype,  debugMode);
crntlogdata.bunsetuMatch = bunsetu_match;
//  オプション　課題　原文　翻訳サポート　メッセージ　などのすべての実験経過を出力
}

Make_advice(sim6.selectSimPoint(simtype), source, target);
//Make_advice の結果としてサポートと各分断片の類似度および問題のある部分ができる（badparts[] に保存）
//また、この中でサポートメッセージを crntlogdata に記録している。

string orgIDs = bns.get_OrgIDs(bunsetu_match.Length);

        if(debugMode)//デバッグモード
{
WordPoint [] orgranking = tool.orgRank(org_try_list,simtype);//並べ替えだけ。候補者はそろってるのが前提。
WordPoint [] orgranking_zero = tool.orgRank(org_zero_list,simtype);
fragData [] fragRanking = tool.fragRank(fragList);
WordPoint [] bunsetuRanking = tool.bunsetuRank(bunsetu_match,simtype);
WordPoint [] diffscoreRanking = tool.diffscoreRank(GoodTreeList,"score");
crntlogdata.orgRanking = bns.IDtoString(orgranking,bunsetu_match);
crntlogdata.org_zeroRanking = bns.IDtoString(orgranking_zero,bunsetu_match);
crntlogdata.fragRanking = bns.IDtoString(fragRanking,bunsetu_match);
crntlogdata.bunsetuRanking = bns.IDtoString(bunsetuRanking,bunsetu_match);
crntlogdata.goodTree = tool.IDtoString_GTD(GoodTreeList, bunsetu_match);
crntlogdata.diffRanking = bns.IDtoString(diffscoreRanking,bunsetu_match);

this.logRichTextBox.Text
+= "\n ▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼\n"
+ repaircount.ToString() + "回目"
+ "\n 課題:\t" + this.kadaiRichTextBox.Text //課題
+ "\n 入力:\t" + crntlogdata.source//入力
+ "\n 翻訳:\t" + crntlogdata.mid//翻訳結果
+ "\n 折返し:\t" + crntlogdata.target//折り返し結果
+ "\n スコア:\t" + Math.Floor(crntlogdata.sim6.selectSimPoint(simtype) * 100)
+ "点（" + crntlogdata.sim6.selectSimPoint(simtype) + ")"//類似度
+ "\n 類似度タイプ:" + tool.printSimType(simtype,simmethod)
+ "\n メッセージ:\t" + crntlogdata.sprtmes //サポートメッセージ
+ "\n サポート:\t" + tool.BadpartsToString(badparts,orgIDs,bunsetu_match)
+ "\n####################ID-文節対応一覧####################:"
+ tool.printIDstring(bunsetu_match)
+ "\n############################################################"
+ "\n ○○○デバッグ用情報○○○"
+ "\n----原文類似ランキング----"
+ tool.WordPointToString(orgranking,bunsetu_match,simtype)
+ "\n------------------------------------------------------------"
+ "\n----1st 原文類似ランキング----"
+ tool.WordPointToString(orgranking_zero,bunsetu_match,simtype)
+ "\n------------------------------------------------------------"
+ "\n----短縮文翻訳適性ランキング----"
+ tool.fragDataToString(fragRanking,bunsetu_match)//これは文節
+ "\n------------------------------------------------------------"
+ "\n----翻訳に適した短縮文リスト----"
+ tool.printGTDList(GoodTreeList,bunsetu_match)
+ "\n------------------------------------------------------------"
+ "\n----DIFF 得点ランキング----"
+ tool.WordPointToString(diffscoreRanking,bunsetu_match,"score")
+ "\n------------------------------------------------------------"
```

```
+ "\n----文節不適ランキング----"
+ tool.WordPointToString(bunsetuRanking,bunsetu_match,"score")
+ "\n ○○○○○○○○○○○○"
+ "\n----------------------オプション設定----------------------"
+ "\n クリアレベル:\t" + (crntlogdata.option.clrsimpoint * 100) + "点"
+ "\n 文節重み:\t" + crntlogdata.option.weightpoint
+ "\n 文断片しきい値:\t" + crntlogdata.option.fragclrpoint
+ "\n ツール:\t" + crntlogdata.option.simmethod
+ "\n デバッグ:ON"
+ "\n 第一キー:" + tool.change(first_key)
+ "\n 第二キー:" + tool.change(second_key)
+ "\n 第三キー:" + tool.change(third_key);
this.logRichTextBox.Select(this.logRichTextBox.Text.Length,0);
this.logRichTextBox.Focus();
this.logRichTextBox.ScrollToCaret();
}
else//非デバッグモード
{
this.logRichTextBox.Text
+= "\n ▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼\n"
+ repaircount.ToString() + "回目"
+ "\n 課題:\t" + this.kadaiRichTextBox.Text //課題
+ "\n 入力:\t" + crntlogdata.source//入力
+ "\n 翻訳:\t" + crntlogdata.mid//翻訳結果
+ "\n 折返し:\t" + crntlogdata.target//折り返し結果
+ "\n スコア:\t" + Math.Floor(crntlogdata.sim6.selectSimPoint(simtype) * 100)
+ "点 (" + crntlogdata.sim6.selectSimPoint(simtype) + ")"//類似度
+ "\n 類似度タイプ:" + tool.printSimType(simtype,simmethod)
+ "\n メッセージ:\t" + crntlogdata.sprtmes //サポートメッセージ
+ "\n サポート:\t" + tool.BadpartsToString(badparts,orgIDs,bunsetu_match)
+ "\n#####################ID-文節対応一覧#####################:"
+ tool.printIDstring(bunsetu_match)
+ "\n############################################################"
+ "\n----------------------オプション設定----------------------"
+ "\n クリアレベル:\t" + (crntlogdata.option.clrsimpoint * 100) + "点"
+ "\n 文節重み:\t" + crntlogdata.option.weightpoint
+ "\n 文断片しきい値:\t" + crntlogdata.option.fragclrpoint
+ "\n ツール:\t" + crntlogdata.option.simmethod
+ "\n デバッグ:OFF"
+ "\n 第一キー:" + tool.change(first_key)
+ "\n 第二キー:" + tool.change(second_key)
+ "\n 第三キー:" + tool.change(third_key);
this.logRichTextBox.Select(this.logRichTextBox.Text.Length,0);
this.logRichTextBox.Focus();
this.logRichTextBox.ScrollToCaret();
}


this.checkmesRichTextBox.Update();
this.logRichTextBox.Update();

repaircount++;

this.Cursor = System.Windows.Forms.Cursors.AppStarting;

if(sim6.selectSimPoint(simtype) < clrsimpoint)Show_badpoint(bunsetu_match);
}
}
this.Cursor = System.Windows.Forms.Cursors.Default;//マウスポインタを元に戻す.
transflg = 0;
threadPointer = null;
}//testtransButton_Click 終了



/// simpoint に応じてアドバイスを出す。ポイントが低いときには Find_badpoint が呼び出される。
private void Make_advice(double simpoint, string org, string back)
{
string advice = "";//スコアの入っていないコメントだけのアドバイスを記録

if (simpoint > 0.9)
```

```
{
this.checkmesRichTextBox.Text = "かなり翻訳に適した文章です．";
advice = this.checkmesRichTextBox.Text;
}

else if (simpoint > clrsimpoint)
{
this.checkmesRichTextBox.Text = "ある程度翻訳に適した文章ですが，もう少し変更した方が良い翻訳結果が得られます．";
advice = this.checkmesRichTextBox.Text;
}

else
{
this.checkmesRichTextBox.Text = "翻訳に不向きな文章です．\n 上の「翻訳サポート」で指摘している箇所を直してみてください。";
//このときだけ、問題のある場所を指摘する。
advice = this.checkmesRichTextBox.Text;
}

            if(simpoint <= clrsimpoint || debugMode)Find_badpoint(org, back);


if(log != null)
{
crntlogdata.badparts = bns.IDtoString(badparts,bunsetu_match);
crntlogdata.sprtmes = advice;//サポートメッセージには得点は含まないようにした
log.logResultList[currentNumber].logDataList.Add(crntlogdata);
}
}

/// 悪い文節を見つけて，直すべきところを指摘
/// 結果（悪い文節）は badparts[] にはいる
private void Find_badpoint(string org, string back)
{
string tmpmid, tmptarget;
string sim;
Similar sim6 = new Similar();
double clrpoint = fragclrpoint;//しきい値 この値は Find_badpoint 内でのみ使われる

if(true)
{
string orgIDs = bns.get_OrgIDs(bunsetu_match.Length);
GoodTreeList.Clear();//一定ポイント以上の類似度を持つ短縮文を記録しておくためのログ。その初期化


string bndnpnsntnc;
foreach ( string IDs in bunsetu_tree)
{
string bundanpensntnc = bns.IDtoString(IDs,bunsetu_match);//文節を ID → string
if (bundanpensntnc.Length == 0) break;
if (bundanpensntnc == org)continue;//得られた短縮文の中で、原文と同じものは短縮文として考えない。

string [] appearIDs;
appearIDs = Regex.Split(IDs," ");
foreach (string appearID in appearIDs){
this.appear[int.Parse(appearID)]++;
}
//短縮文ごとに類似度を計算
tmpmid = translation.Translate(bundanpensntnc, "J", "E");
tmptarget = translation.Translate(tmpmid, "E", "J");

tmpmid = Regex.Replace(tmpmid, "\r\n", "");
tmptarget = Regex.Replace(tmptarget, " |  |\r\n", "");

// CS 研のツールは半角が使えないので，全角に変換する必要がある．
if (simmethod.Equals("CS"))
{
bndnpnsntnc = Trans_hankaku_zenkaku(bundanpensntnc);
tmptarget = Trans_hankaku_zenkaku(tmptarget);
}

if (bundanpensntnc.Equals("") || tmptarget.Equals(""))
```

```
{
sim6 = sim6.setSimPoint("0 0 0 0 0 0");
}
else
{
sim = similar.Cal_similar(bundanpensntnc, tmptarget, simmethod, hostName, hostPort);//類似度計算
sim6 = tool.setSimPoint(sim);
}

//この時点で断片ごとに折り返し後との類似度ができている。
//simpoint:類似度　bundanpensntnc:短縮文, tmpmid:短縮文の英文, tmptarget:短縮文の折り返し
// log に文断片を書き出し

fragList.Add(new fragData(0, IDs, tmpmid, tmptarget, sim6,simtype));
//fragList はデバッグモードでなくても使う可能性があるため、下の if の中から飛び出させた


if(debugMode || (sim6.selectSimPoint(simtype) > clrpoint))//デバッグモードか閾値よりも類似度が高いときだけ実行
{
string temp_sim;//一時退避場所
Similar temp_sim6 = new Similar();
Similar sim_org_zero6 = new Similar();//断片と一回目の原文との類似度
Similar sim_org_try6 = new Similar();//断片と原文との類似度
string [] problem;

temp_sim = similar.Cal_similar(org_zero, bundanpensntnc, simmethod, hostName, hostPort);//類似度計算
temp_sim6 = tool.setSimPoint(temp_sim);//類似度が 6 つでも 1 つでもここで処理する。
sim_org_zero6 = temp_sim6.makeCopy();
temp_sim = similar.Cal_similar(org, bundanpensntnc, simmethod, hostName, hostPort);//類似度計算
temp_sim6 = tool.setSimPoint(temp_sim);

sim_org_try6 = temp_sim6.makeCopy();

//////////////////////////////原文 1st 原文との類似度を計算完了//////////////////////////////////

//原文から優秀な短縮文を切り取ったものをつくる。


if(debugMode)//デバッグ情報のための準備
{
org_try_list.Add(new WordPoint(IDs,sim_org_try6));//原文類似ランキング用リスト
org_zero_list.Add(new WordPoint(IDs,sim_org_zero6));//1st 原文類似ランキング用リスト
}

if(sim6.selectSimPoint(simtype) > clrpoint)//閾値よりも類似度が高ければその短縮文を記憶
{
                            problem = tool.GetDiff(orgIDs,IDs);
foreach(string proTemp in problem){
this.point[int.Parse(proTemp)]++;
}
GoodTreeData GTD = new GoodTreeData(IDs,sim6,sim_org_zero6,sim_org_try6,
problem,first_key, second_key, third_key,simtype);
//文断片　自分との類似度　一回目と断片との類似度　現在のものと断片との類似度 優秀な断片をとり除いた分の文節集合
//ソートに関して、第一キー　第二キー　第三キー
GoodTreeList.Add(GTD);//ここに優秀な文断片を保管↑の GoodTreeData を持っている。
}
}
}//ここですべての短縮文に対して M の生成が完了　ソートはまだ

GoodTreeList.Sort();//ここで ID
GoodTreeList.Reverse();
tool.giveWeight(GoodTreeList,this.weightpoint,6);//GoodTreeList の文節に重みをつける

//以下翻訳劣化の原因となる文節にポイントをつけていく。
int i;
this.APfrac = new double [this.appear.Length];
for(i=0;i<this.appear.Length;i++){
this.APfrac[i] = this.point[i]/this.appear[i];
}

                int array_index=0;
```

```
int array_size = GoodTreeList.Count;
double max = 0;

int orgID_num = Regex.Split(orgIDs," ").Length;
for(i=0;i <orgID_num;i++)bunsetu_match[i].score=0;

//問題があると思しき文節の重みを計算する。
for(array_index=0;array_index<array_size;array_index++)
{
if(array_index == array_size)break;

GoodTreeData GTDtemp = (GoodTreeData)GoodTreeList[array_index];
//GTD の List からデータをひとつ取り出す
string [] problems = GTDtemp.GetProblem();
//problems は問題のあると思われる文節の ID 集合
foreach(string problem_temp in problems)//問題のある文節一つ一つに対して重みを足しこんでいく
{
bunsetu_match[int.Parse(problem_temp)].score =
bunsetu_match[int.Parse(problem_temp)].score + GTDtemp.point * this.APfrac[int.Parse(problem_temp)];
if(bunsetu_match[int.Parse(problem_temp)].score > max) max = bunsetu_match[int.Parse(problem_temp)].score;
}
}
                //ここまでで文節ごとのポイント計算終了。
//スコアが最も高い ID を取り出して、temp に書き込んでいく。
string temp = "";
for(i=0;i <orgID_num;i++)
{
if(bunsetu_match[i].score == max){
if(temp=="")temp = i.ToString();
else temp = temp + " " + i.ToString();
}
}

if(max == 0)//ポイントが全て 0 のときにはさらに別の処理をする。
{
temp = tool.findSecondBadpoint(fragList,clrpoint,orgIDs,bunsetu_match,simmethod,hostName,hostPort);
}

badparts_length = Regex.Split(temp," ").Length;
badparts = temp;//badparts[] にもっとも問題があると思しき文節群を保管
}
}//Find_badpoint の終了


/// ここで文字に色をつける。その対象は badparts[] に入っている文字列
private void Show_badpoint(bns [] bunsetu_match)
{
if(badparts == "")
{
this.highlight.Text = "翻訳に適さない箇所を発見できませんでした。";
return;
}
string text = Regex.Replace(this.sourceRichTextBox.Text, " |  |\r\n|\n", "");
this.highlight.Text = text;
string [] badtexts = Regex.Split(badparts," ");
foreach (string badtext in badtexts)//badtext には文節一つの ID が string 型で入っている。
{
int address = tool.get_ID_address(badtext,bunsetu_match);//ID が指定する文節の先頭のアドレスを獲得
string badbns = bns.IDtoString(badtext,bunsetu_match);//ID が指定する文節のを string 型にしたもの。

this.highlight.SelectionStart = address;
this.highlight.SelectionLength=badbns.Length;
this.highlight.SelectionBackColor = System.Drawing.Color.Aqua;
}
return;
}

private void submitButton_Click(object sender, System.EventArgs e)
{
if(MessageBox.Show(
"この文章の翻訳で，良いですか？\n\n" +
```

```
"  修正後の文章\n\n" +
this.sourceRichTextBox.Text + "\n\n",
"確認",System.Windows.Forms.MessageBoxButtons.OKCancel)
== DialogResult.OK)
{
Questions.InQuestion(log, currentNumber);
if(LoadSampleText(++currentNumber))
{//次の問題に行くための準備。ここで false が返ってきたら今の問題が最終問題ってこと
//初期化
org_zero = "";
badparts = "";
GoodTreeList.Clear();

this.checkmesRichTextBox.Text = "ここには，より良い翻訳を行う為のメッセージが表示されます．";
this.highlight.Clear();
this.highlight.Text = "ここには，原文の修正個所候補を提示します．";
repaircount = 1;
if(log != null)
{
log.logResultList.Add(new LogResult(currentNumber));
}
}
else
{
Questions.PostQuesiton(log);
MessageBox.Show("ありがとうございました");
this.Close();
}
}
}


private void cutsourceMenuItem_Click(object sender, System.EventArgs e)
{
if (sourceRichTextBox.SelectionLength > 0) sourceRichTextBox.Cut();
}

private void pastesourceMenuItem_Click(object sender, System.EventArgs e)
{
if (Clipboard.GetDataObject().GetDataPresent(DataFormats.Text))
{
sourceRichTextBox.Paste();
}
}

private void cpsourceMenuItem_Click(object sender, System.EventArgs e)
{
if (sourceRichTextBox.SelectionLength > 0) sourceRichTextBox.Copy();
}

private void cpkadaiMenuItem_Click(object sender, System.EventArgs e)
{
if (kadaiRichTextBox.SelectionLength > 0) kadaiRichTextBox.Copy();
}

private void cpcheckmesMenuItem_Click(object sender, System.EventArgs e)
{
if (checkmesRichTextBox.SelectionLength > 0) checkmesRichTextBox.Copy();
}

private void cplogMenuItem_Click(object sender, System.EventArgs e)
{
if (logRichTextBox.SelectionLength > 0) logRichTextBox.Copy();
}

/// プログラムが終了するときにはログを"ユーザ名.xml"に保存する
private void SimtransForm_Closed(object sender, System.EventArgs e)
{
LogTool.WriteLog(log, username + ".xml");
}
```

```
private void configButton_Click(object sender, System.EventArgs e)
{
SetOption();
}


public string Trans_hankaku_zenkaku(string src)
{
Regex r = new Regex(".+?[。?! \\.\\?!]+");
src = Regex.Replace(src, "\r\n", "");


src = Regex.Replace(src, "-", "－");
src = Regex.Replace(src, " ", "　");
src = Regex.Replace(src, "!", "！");
src = Regex.Replace(src, "\"", "”");
src = Regex.Replace(src, "#", "＃");
src = Regex.Replace(src, "%", "％");
src = Regex.Replace(src, "&", "＆");
src = Regex.Replace(src, "'", "’");
src = Regex.Replace(src, "0", "０");
src = Regex.Replace(src, "1", "１");
src = Regex.Replace(src, "2", "２");
src = Regex.Replace(src, "3", "３");
src = Regex.Replace(src, "4", "４");
src = Regex.Replace(src, "5", "５");
src = Regex.Replace(src, "6", "６");
src = Regex.Replace(src, "7", "７");
src = Regex.Replace(src, "8", "８");
src = Regex.Replace(src, "9", "９");
src = Regex.Replace(src, ":", "：");
src = Regex.Replace(src, ";", "；");
src = Regex.Replace(src, "<", "＜");
src = Regex.Replace(src, "=", "＝");
src = Regex.Replace(src, ">", "＞");
src = Regex.Replace(src, "@", "＠");
src = Regex.Replace(src, "A", "Ａ");
src = Regex.Replace(src, "B", "Ｂ");
src = Regex.Replace(src, "C", "Ｃ");
src = Regex.Replace(src, "D", "Ｄ");
src = Regex.Replace(src, "E", "Ｅ");
src = Regex.Replace(src, "F", "Ｆ");
src = Regex.Replace(src, "G", "Ｇ");
src = Regex.Replace(src, "H", "Ｈ");
src = Regex.Replace(src, "I", "Ｉ");
src = Regex.Replace(src, "J", "Ｊ");
src = Regex.Replace(src, "K", "Ｋ");
src = Regex.Replace(src, "L", "Ｌ");
src = Regex.Replace(src, "M", "Ｍ");
src = Regex.Replace(src, "N", "Ｎ");
src = Regex.Replace(src, "O", "Ｏ");
src = Regex.Replace(src, "P", "Ｐ");
src = Regex.Replace(src, "Q", "Ｑ");
src = Regex.Replace(src, "R", "Ｒ");
src = Regex.Replace(src, "S", "Ｓ");
src = Regex.Replace(src, "T", "Ｔ");
src = Regex.Replace(src, "U", "Ｕ");
src = Regex.Replace(src, "V", "Ｖ");
src = Regex.Replace(src, "W", "Ｗ");
src = Regex.Replace(src, "X", "Ｘ");
src = Regex.Replace(src, "Y", "Ｙ");
src = Regex.Replace(src, "Z", "Ｚ");
src = Regex.Replace(src, "a", "ａ");
src = Regex.Replace(src, "b", "ｂ");
src = Regex.Replace(src, "c", "ｃ");
src = Regex.Replace(src, "d", "ｄ");
src = Regex.Replace(src, "e", "ｅ");
src = Regex.Replace(src, "f", "ｆ");
src = Regex.Replace(src, "g", "ｇ");
src = Regex.Replace(src, "h", "ｈ");
src = Regex.Replace(src, "i", "ｉ");
```

```
src = Regex.Replace(src, "j", "ｊ");
src = Regex.Replace(src, "k", "ｋ");
src = Regex.Replace(src, "l", "ｌ");
src = Regex.Replace(src, "m", "ｍ");
src = Regex.Replace(src, "n", "ｎ");
src = Regex.Replace(src, "o", "ｏ");
src = Regex.Replace(src, "p", "ｐ");
src = Regex.Replace(src, "q", "ｑ");
src = Regex.Replace(src, "r", "ｒ");
src = Regex.Replace(src, "s", "ｓ");
src = Regex.Replace(src, "t", "ｔ");
src = Regex.Replace(src, "u", "ｕ");
src = Regex.Replace(src, "v", "ｖ");
src = Regex.Replace(src, "w", "ｗ");
src = Regex.Replace(src, "x", "ｘ");
src = Regex.Replace(src, "y", "ｙ");
src = Regex.Replace(src, "z", "ｚ");
src = Regex.Replace(src, "{", " {");
src = Regex.Replace(src, "}", "} ");
src = Regex.Replace(src, "~", "〜");
return src;
}

private void checkmesRichTextBox_TextChanged(object sender, System.EventArgs e)
{}

private void label1_Click(object sender, System.EventArgs e)
{}

private void transCancel_Click(object sender, System.EventArgs e)
{
if((transflg==1) && (threadPointer!=null))
{
threadPointer.Abort();
this.checkmesRichTextBox.Text = "ここには，より良い翻訳を行う為のメッセージが表示されます．";
this.Cursor = System.Windows.Forms.Cursors.Default;
threadPointer = null;
transflg = 0;
}
return;
}

private void kadaiRichTextBox_TextChanged(object sender, System.EventArgs e)
{}
}
}
```

A.2.2 Translation.cs

```
using System;
using System.Text.RegularExpressions;
using System.Collections;
using System.Web;
using System.Net;
using System.IO;

namespace SimTrans
{
public class Translation
{
Hashtable cache;

public Translation()
{
cache = new Hashtable();
}

public string Translate( string text, string source, string target )
{
string sentences = "";
Regex r = new Regex(".+?[。?! \\.\\?!]+");//"+?"は"一回以上の繰り返し"を意味する
```

```
if (source.Equals("J"))
{
text = Regex.Replace(text, " |　|\n|\r\n", "");//日本語を翻訳するときにはスペースと改行を消す
}
else
{
text = Regex.Replace(text, " |　|\n|\r\n", " ");//日本語以外を翻訳するときには全角スペースと改行を半角スペースにする
}

MatchCollection matches = r.Matches(text);
foreach(Match match in matches)
{
sentences += TranslateSentence(match.Value, source, target);
}
return sentences;
}

private string TranslateSentence( string sentence, string source, string target )
{
if( cache.ContainsKey(sentence.GetHashCode()) )//既に翻訳したことがあるかをハッシュで見る
{
return (string)cache[sentence.GetHashCode()];//既に翻訳したことがあればいちいちサーバーに送らない.
}
else
{
string result = TranslateViaHttp(sentence, source, target);
cache.Add(sentence.GetHashCode(), result);
return result;
}
}

private string TranslateViaHttp( string sentence, string source, string target)
{
const string URL = "http://ice.kuis.kyoto-u.ac.jp:8081/ICETranslator/ICETranslator";
string parameter = "source=" + source + "&target=" + target + "&service=J&text="
+ HttpUtility.UrlEncode(sentence, System.Text.Encoding.UTF8);

HttpWebRequest req = (HttpWebRequest)HttpWebRequest.Create(URL);
req.Proxy = GlobalProxySelection.Select;
req.KeepAlive = true;
req.Method = "POST";
req.ContentLength = parameter.Length;
req.ContentType = "application/x-www-form-urlencoded";

StreamWriter output = null;
try
{
output = new StreamWriter(req.GetRequestStream());
output.Write(parameter);
}
catch(Exception e)
{
return "";
}
finally
{
if(output != null)
{
output.Close();
}
}

StreamReader input = null;
try
{
HttpWebResponse res = (HttpWebResponse)req.GetResponse();
input = new StreamReader(res.GetResponseStream());
return input.ReadToEnd();
}
catch(Exception e)
```

```
{
return "";
}
finally
{
if(input != null)
{
input.Close();
}
}
}
}
}
```

A.2.3 ToolFunction.cs

```
//#define DEBUG
//#undef DEBUG
//#define RELEASE
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.IO;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading;
namespace SimTrans
{
public class ToolFunction
{
public ToolFunction()
{
}

public string [] GetDiff(string orgIDs, string IDs)
{
string ret0 = "";

string [] org_temp = Regex.Split(orgIDs, " ");
string [] str_temp = Regex.Split(IDs, " ");

foreach(string one_temp in org_temp)
{
int flg=0;
foreach(string two_temp in str_temp)
{
if(one_temp.Equals(two_temp))flg=1;
}
if(flg==0)
{
if(ret0.Equals(""))
{
ret0 = one_temp;
}
else
{
ret0 = ret0 + " " + one_temp;
}
}
}
string [] ret = Regex.Split(ret0, " ");
return ret;
}

/// badparts を括弧でかこった文を作る
public string BadpartsToString(string badparts, string orgIDs, bns [] bunsetu_match)
{
string ret = "";
string [] badtemps = Regex.Split(badparts," ");
```

```
string [] orgID = Regex.Split(orgIDs," ");
badtemps = Regex.Split(badparts," ");
int flg = 0;
foreach(string ID in orgID)
{
flg =0;
foreach(string badtemp in badtemps)
{
if(badtemp == ID)flg =1;
}

if(flg==0){ret = ret + bunsetu_match[int.Parse(ID)].bunsetu;}
if(flg==1){ret = ret + "(" + bunsetu_match[int.Parse(ID)].bunsetu + ")";}
//問題のある文節は括弧を付けて出力
}
return ret;

}

// 短縮文に重みを付与する
public void giveWeight(ArrayList al, double weight, int count)
{
if(al == null)return;
GoodTreeData temp = new GoodTreeData();
GoodTreeData before = new GoodTreeData();
int i = 0;
int now = 0;

//j の値によって上位何番目の断片にポイントがつくかを決める。
if(al.Count > count)//何番までランク付けするか　<　対象の数
{
while(i < count || now < count)
{
temp = ((GoodTreeData)al[i]).makeCopy();
if(temp.get_similarity(temp.first,temp.simtype) == before.get_similarity(before.first,before.simtype)
&& (temp.get_similarity(temp.second,temp.simtype) == before.get_similarity(before.second,before.simtype))
&& (temp.get_similarity(temp.third,temp.simtype) == before.get_similarity(before.third,before.simtype))
 )
{
temp.point = before.point;
//同じ値を持つものには同じ重みをつける。
now = now;
}
else
{
temp.point = (count - i) * weight;
now = i;
if(now >= count)break;
}
al[i] = temp;
before = temp;
i++;
}

while(i < al.Count)
{//残りのやつ（ランク外）の短縮文にはスコアとして 0 を与える。
temp = (GoodTreeData)al[i];
temp.point = 0;
al[i] = temp;
i++;
}

}
else
{
while(i < al.Count)
{
temp = ((GoodTreeData)al[i]).makeCopy();
if(temp.get_similarity(temp.first,temp.simtype) == before.get_similarity(before.first,before.simtype)
&& (temp.get_similarity(temp.second,temp.simtype) == before.get_similarity(before.second,before.simtype))
&& (temp.get_similarity(temp.third,temp.simtype) == before.get_similarity(before.third,before.simtype))
```

```
)  temp.point = before.point;
//同じ値を持つものには同じ重みをつける。
else temp.point = (al.Count - i) * weight;
al[i] = temp;
before = temp;
i++;
}
}
return;
}

// 文節に順位を付けたものを返す
public WordPoint [] bunsetuRank(bns [] bunsetu_match, string simtype)
{
ArrayList al = new ArrayList();
WordPoint before = new WordPoint();
WordPoint [] ret = new WordPoint [bunsetu_match.Length];
WordPoint wptemp;
int i = 0;
while(i < bunsetu_match.Length)
{
                wptemp = new WordPoint();
wptemp.score = bunsetu_match[i].score;//ID に与えられたスコア
wptemp.word = i.ToString();//ID
wptemp.rank = -1;
wptemp.simtype = "score";

al.Add(wptemp);
i++;
}
al.Sort();
al.Reverse();
i = 0;
before.score = -1;
while(i <bunsetu_match.Length)
{
ret[i] = ((WordPoint)al[i]).makeCopy();
if(ret[i].score == before.score) ret[i].rank = before.rank;
else ret[i].rank = i+1;
before = ret[i].makeCopy();
i++;
}
return ret;
}


public string printGTDList(ArrayList GoodTreeList, bns [] bunsetu_match)
{
int i =0;
int rank = 0;
string ret = "";
GoodTreeData temp = new GoodTreeData();
GoodTreeData before = new GoodTreeData();

while(i < GoodTreeList.Count)
{
temp = ((GoodTreeData)GoodTreeList[i]).makeCopy();
if(temp.get_similarity(temp.first,temp.simtype) == before.get_similarity(before.first,before.simtype)
&& (temp.get_similarity(temp.second,temp.simtype) == before.get_similarity(before.second,before.simtype))
&& (temp.get_similarity(temp.third,temp.simtype) == before.get_similarity(before.third,before.simtype))
) rank = rank;
else rank = i;
ret = ret + "\n"
+ (rank + 1) + ":[" + bns.IDtoString(temp.sentence,bunsetu_match) + "]" + "\n"
+ "\tpnt=" + temp.point + "\n"
+ "\tdiff:" + bns.IDtoString((box2line(temp.problem)),bunsetu_match) + "\n"
+ "\ttry_similarity=" + temp.get_similarity("try_sim",temp.simtype) + "\n"
+ "\torg_similarity=" + temp.get_similarity("org_sim",temp.simtype) + "\n"
+ "\tself_similarity=" + temp.get_similarity("self_sim",temp.simtype) ;
before = temp;
i++;
```

```
}
return ret;
}


public WordPoint [] diffscoreRank(ArrayList gtl, string simtype)//GoodTreeList
{
WordPoint temp = new WordPoint();
WordPoint before = new WordPoint();
GoodTreeData gtd = new GoodTreeData();
WordPoint [] ret = new WordPoint[gtl.Count];
int i = 0;
string [] box;
string str  = "";

while(i <gtl.Count)
{
temp = new WordPoint();
gtd = (GoodTreeData)gtl[i];
box = gtd.problem;
str = box2line(box);
temp.word = str;
temp.score = gtd.point;//与えられた重み
temp.simtype = "score";

ret[i] = temp;

if(ret[i].score ==before.score) ret[i].rank = before.rank;
else ret[i].rank = i+1;

before = ret[i];
i++;
}
return ret;
}


// string 配列の内容を一つの string にする。
private string box2line(string [] hoge)
{
if(hoge == null)return "";
int num = hoge.Length;
int i = 0;
string ret = "";
while(i < hoge.Length)
{
if(ret == "")ret = hoge[i];
else
{
ret = ret + " " + hoge[i];
}
i++;
}
return ret;
}

public string change (string str)
{
if(str == "try_sim")return "原文類似度";
if(str == "org_sim")return "1st 原文類似度";
if(str == "self_sim")return "折返し文との類似度";
return "エラーです。";
}


public WordPoint [] orgRank (ArrayList al, string simtype)//al は WordPoint が入っているリスト。
{
WordPoint [] ret = new WordPoint [al.Count];
WordPoint before = new WordPoint();

foreach (WordPoint  tempWP in al){
```

```
tempWP.simtype = simtype;
}

al.Sort();
al.Reverse();

int i = 0;
while(i < al.Count)
{
ret[i] = ((WordPoint)al[i]).makeCopy();
if(ret[i].sim6.selectSimPoint(simtype) == before.sim6.selectSimPoint(simtype))
{
ret[i].rank = before.rank;
}
else
{
ret[i].rank = i+1;
}

before = ret[i];
i++;
}
return ret;
}


/// ランク : 文字列 :: 類似度
public string WordPointToString(WordPoint [] wp, bns [] bunsetu_match, string simtype)
{
string ret = "";
int i = 0;
string tempID;
while (i < wp.Length)
{
tempID = wp[i].getWord();
ret = ret + "\n" + wp[i].rank + ":" + bns.IDtoString(tempID,bunsetu_match) + "::" + wp[i].getSimpoint(simtype);
i++;
}
return ret;
}

public fragData [] fragRank(ArrayList al)
{
fragData [] ret = new fragData [al.Count];
fragData before = new fragData();

al.Sort();
al.Reverse();
int i = 0;
while(i < al.Count)
{
ret[i] = ((fragData)al[i]).makeCopy();
if(ret[i].selectSimpoint(ret[i].simtype) == before.selectSimpoint(before.simtype))ret[i].rank = before.rank;
else ret[i].rank = i+1;
before = ret[i];
i++;
}
return ret;
}

/// fragdata(org english back point rank からなる) を表示させるために一つの string にする
public string fragDataToString(fragData [] data,bns [] bunsetu_match)
{
string ret = "";
int i =0;
while(i < data.Length)
{
ret = ret + "\n" + data[i].rank + ":" + bns.IDtoString(data[i].org,bunsetu_match)
+ "\n" + "\tEng=" + data[i].english
+ "\n" + "\tBck=" + data[i].back
+ "\n" + "\tPnt=" + data[i].selectSimpoint(data[i].simtype);
```

```
i++;
}
return ret;
}

public bool include(string str, LastWord lw)
{
str = Regex.Replace(str,"、|。","");
if(str.IndexOf(lw.word) == -1)return false;
if((str.LastIndexOf(lw.word) + lw.word.Length) == str.Length)return true;
else return false;
        }

public string findSecondBadpoint(ArrayList al, double clrpoint,string orgIDs,
bns [] bunsetu_match, string simmethod, string hostName, int hostPort)
{
//全ての短縮文の短-折類似度が clrpoint よりも低いってことが前提
//al には fragdata 型のデータ (rank orgID eng back simpoint) が入っている。
double clr2 = clrpoint;
ArrayList tempList = new ArrayList();
fragData fragTemp;
double [] IDpoint = new double [Regex.Split(orgIDs," ").Length];
int i =0;
int past =0;

while(clr2>0 && tempList.Count<4)
{
i=0;
past = tempList.Count;
tempList.Clear();
clr2 = clr2 - 0.05;
while(i < al.Count)
{
fragTemp = new fragData();
fragTemp = (fragData)al[i];
if(fragTemp.english!=""
&& fragTemp.selectSimpoint(fragTemp.simtype) >= clr2
&& fragTemp.selectSimpoint(fragTemp.simtype) < clrpoint)
{
tempList.Add(fragTemp);//下げたしきい値を超えたやつから fragTemp に追加していく。
}
i++;
}
if(past!=0 && ((tempList.Count/past) >2))break;
}
//ここまでで、下げたしきい値を超えたいくつかの短縮文の fragdata が tempList に入っている

for(i=0;i<IDpoint.Length;i++) IDpoint[i] = 0;//文節の初期化

double max = 0;
i=0;
string [] bunsetuIDs;//短縮文をあらわす ID セットを一時的に入れておく string
while(i < tempList.Count)//それぞれの短縮文について以下の操作を行う
{
fragTemp = new fragData();
fragTemp = (fragData)tempList[i];
bunsetuIDs = Regex.Split(fragTemp.org, " ");
foreach(string bunsetuID in bunsetuIDs)
{
IDpoint[int.Parse(bunsetuID)] = IDpoint[int.Parse(bunsetuID)] + 1;
if(IDpoint[int.Parse(bunsetuID)] > max) max = (int)IDpoint[int.Parse(bunsetuID)];
}
i++;
}

if(max == 0)return "";

string ret = "";
i=0;
for(i=0;i<IDpoint.Length;i++){//ポイントが最も高くついた形態素を取り出す。
if(IDpoint[i] == max)
```

```
{
if(ret == "")//先頭のとき
{
ret = i.ToString();
}
else//先頭以外のとき
{
ret = ret + " " + i.ToString();
}
}
}

return ret;
}

/// ID で指定された文節の始まるアドレスを返す
public int get_ID_address(string ID,bns [] bunsetu_match)
{
if(int.Parse(ID) == 0)return 0;
string prestring = "";
int i;
for(i=0;i<int.Parse(ID);i++)
{
prestring = prestring + bunsetu_match[i].bunsetu;
}
return prestring.Length;
}

public string printIDstring(bns [] bunsetu_match){
string ret = "";
if(bunsetu_match.Length == 0)return ret;

int i;
for(i=0;i<bunsetu_match.Length;i++){
ret = ret + "\n" + i.ToString() + ":" + bunsetu_match[i].bunsetu ;
}
return ret;
}

public Similar setSimPoint(string sim)
{
            Similar ret = new Similar();
if(sim.Length > 6)
{
string [] temp = Regex.Split(sim," ");
ret.precision = double.Parse(temp[0]);//類義語を考慮。
ret.recall = double.Parse(temp[1]);//類義語を考慮。
ret.f_value = double.Parse(temp[2]);//類義語を考慮。
ret.precision_zero = double.Parse(temp[3]);
ret.recall_zero =  double.Parse(temp[4]);
ret.f_value_zero =  double.Parse(temp[5]);
return ret;
}
else{
ret.recall = double.Parse(sim);
}
return ret;
}


public string printSimType(string simtype,string simmethod){
if(simmethod == "BLEU" || simmethod == "NIST")return "";
switch(simtype)
{
case "precision":
return "precision(類義語を考慮)";

case "recall":
return "recall(類義語を考慮)";

case "f-value":
```

```
return "f 値 (類義語を考慮)";

case "precision_zero":
return "precision(類義語を考慮せず)";

case "recall_zero":
return "recall(類義語を考慮せず)";

case "f-value_zero":
return "f 値 (類義語を考慮せず)";

default:
return "";
}
}

public GoodTreeData [] IDtoString_GTD(ArrayList al, bns [] bunsetu_match)
{
GoodTreeData [] ret = new GoodTreeData [al.Count];
GoodTreeData temp;
int i,j;
for(i=0;i<al.Count;i++)
{
temp = new GoodTreeData();
temp = ((GoodTreeData)al[i]).makeCopy();
temp.sentence = bns.IDtoString(temp.sentence,bunsetu_match);
for(j=0;j<(temp.problem).Length;j++)
{
temp.problem[j]  = bns.IDtoString(temp.problem[j],bunsetu_match);
}
ret[i] = temp;
}
return ret;
}
}//ここまでで ToolFunction クラスは終了

//////////////////////ここから下はクラス定義//////////////////////////
public class fragData : IComparable
{
public int rank;
public string org;
public string english;
public string back;
public Similar sim6;
public string simtype;

public fragData()
{
this.rank = 0;
this.org = "";
this.english = "";
this.back = "";
this.sim6 = new Similar();
this.simtype = "";
}
public fragData(int rank, string original, string eng, string backed, Similar sim6,string simtype)
{
this.rank = rank;
this.org = original;
this.english = eng;
this.back = backed;
this.sim6 = sim6.makeCopy();
this.simtype = simtype;
}
public fragData makeCopy(){
fragData ret = new fragData();

ret.org = this.org;
ret.english = this.english;
ret.back = this.back;
ret.sim6 = this.sim6.makeCopy();
```

```
ret.simtype = this.simtype;
ret.rank =this.rank;

return ret;
  }
public double selectSimpoint(string simtype){
if(simtype == "precision") return this.sim6.precision;//類義語を考慮。
if(simtype == "recall") return this.sim6.recall;//類義語を考慮。
if(simtype == "f-value") return this.sim6.f_value;//類義語を考慮。
if(simtype == "precision_zero") return this.sim6.precision_zero;
if(simtype == "recall_zero") return this.sim6.recall_zero;
if(simtype == "f-value_zero") return this.sim6.f_value_zero;
return -2;
}
public int CompareTo(object obj)//obj:呼び出されたオブジェクト
{
if(obj == null)return 1;

if (obj is fragData)
{
fragData other = (fragData)obj;
if(this.selectSimpoint(simtype) > other.selectSimpoint(simtype))return 1;
if(this.selectSimpoint(simtype) < other.selectSimpoint(simtype))return -1;
return 0;
}
throw new ArgumentException("object is not a fragData");
}
}


public class LastWord
{
public string word;
public int good;
public int bad;
public LastWord()
{
this.word = "";
this.good = 0;
this. bad = 0;
}
}


public class WordPoint : IComparable
{
public int rank;
public string word;
        public string simtype;
public Similar sim6;

public double score;

public WordPoint()
{
rank = 0;
word = "default";
sim6 = new Similar();
simtype = "";
score = 0;
}
public WordPoint(string str)
{
rank = 0;
word = str;
sim6 = new Similar();
simtype = "";
score = 0;
}
public WordPoint(string str, Similar sim)
{
```

```
rank = 0;
word =str;
sim6 = new Similar();
sim6 = sim;
simtype = "";
score = 0;
}
public WordPoint(int num, string str, Similar sim)
{
rank = num;
word =str;
sim6 = new Similar();
sim6 = sim;
simtype = "";
score = 0;
}
public double getSimpoint(string simtype)
{
if(simtype == "precision") return this.sim6.precision;//類義語を考慮。
if(simtype == "recall") return this.sim6.recall;//類義語を考慮。
if(simtype == "f-value") return this.sim6.f_value;//類義語を考慮。
if(simtype == "precision_zero") return this.sim6.precision_zero;
if(simtype == "recall_zero") return this.sim6.recall_zero;
if(simtype == "f-value_zero") return this.sim6.f_value_zero;
if(simtype == "score") return this.score;
return -2;
}

public WordPoint makeCopy(){
WordPoint ret = new WordPoint();
ret.rank = this.rank;
ret.score = this.score;
ret.sim6 = this.sim6.makeCopy();
ret.simtype = this.simtype;
ret.word = this.word;
return ret;
}

public int getRank()
{
return this.rank;
}

public string getWord()
{
return this.word;
}


public int CompareTo(object obj)//obj:呼び出されたオブジェクト
{
if(obj == null)return 1;

if (obj is WordPoint)
{
WordPoint other = (WordPoint)obj;
if(this.getSimpoint(simtype) > other.getSimpoint(simtype))return 1;
if(this.getSimpoint(simtype) < other.getSimpoint(simtype))return -1;
return 0;
}
throw new ArgumentException("object is not a WordPoint");
}
}
}
```

A.2.4 Similar.cs

```
using System;
using System.Collections;
using System.Text.RegularExpressions;
using System.Web;
using System.Net;
```

```
using System.Net.Sockets;
using System.IO;

namespace SimTrans
{
public class Similar
{
public double precision;
public double recall;
public double f_value;

public double precision_zero;
public double recall_zero;
public double f_value_zero;


public Similar()
{
this.precision=-1;
this.recall=-1;
this.f_value=-1;

this.precision_zero=-1;
this.recall_zero=-1;
this.f_value_zero=-1;
}

public double selectSimPoint(string simtype)
{
if(simtype == "precision") return this.precision;//類義語を考慮。
if(simtype == "recall") return this.recall;//類義語を考慮。
if(simtype == "f-value") return this.f_value;//類義語を考慮。
if(simtype == "precision_zero") return this.precision_zero;
if(simtype == "recall_zero") return this.recall_zero;
if(simtype == "f-value_zero") return this.f_value_zero;
return -2;
}

public Similar setSimPoint(string sim)
{
Similar ret = new Similar();
if(sim.Length > 6)
{
string [] temp = Regex.Split(sim," ");
ret.precision = double.Parse(temp[0]);//類義語を考慮。
ret.recall = double.Parse(temp[1]);//類義語を考慮。
ret.f_value = double.Parse(temp[2]);//類義語を考慮。
ret.precision_zero = double.Parse(temp[3]);
ret.recall_zero =  double.Parse(temp[4]);
ret.f_value_zero =  double.Parse(temp[5]);
return ret;
}
else
{
ret.recall = double.Parse(sim);
}
return ret;
}

/// <summary>
/// 自分のコピーを新しくつくって出力
/// </summary>
/// <returns></returns>
public Similar makeCopy()
{
Similar ret = new Similar();
ret.precision = this.precision;
ret.recall = this.recall;
ret.f_value = this.f_value;
ret.precision_zero = this.precision_zero;
ret.recall_zero = this.recall_zero;
```

```
ret.f_value_zero = this.f_value_zero;
return ret;
}

public string Cal_similar(string org, string back, string simmethod, string hostName, int hostPort)
{
string URL = "";
int portnum = 80;
string parameter = "";
org = Regex.Replace(org, ",| |  |\r\n|\n", "");
back = Regex.Replace(back, ",| |  |\r\n|\n", "");

if((hostPort != 8080 && hostPort != 80) && simmethod == "SELF")//TCP 通信を自分で設定するとき。
{
Socket simsocket = null;
byte[] bytes = new byte[1024];
byte[] recbytes = new byte[1024];

System.Net.IPHostEntry iphe = System.Net.Dns.Resolve(hostName);
ipad = iphe.AddressList[0];//IP アドレス決定
portnum = hostPort;

parameter = "org=" + org + "&back=" + back + "&callmethod=similar\n";
bytes = System.Text.Encoding.UTF8.GetBytes(parameter);
IPEndPoint ipe = new IPEndPoint(ipad, portnum);
string tmps;

try //http でないときはここで翻訳してもらう文字列を依頼先に送る
{
simsocket = new Socket(ipe.AddressFamily, SocketType.Stream, ProtocolType.Tcp);
simsocket.Connect(ipe);

if(simsocket.Connected)
{

Console.WriteLine("connected:");
simsocket.Send(bytes, 0, bytes.Length,0);
simsocket.Receive(recbytes);
tmps = System.Text.Encoding.UTF8.GetString(recbytes,0,recbytes.Length);
return(tmps);
}

}
catch(SocketException e)
{
return e.Message;
}
catch(Exception e)
{
return e.Message;
}
finally
{
if(simsocket != null)
{
simsocket.Close();
}
}
}
else //ポート番号が 80 or 8080 か、SELF に設定されていないとき。http で接続する。
{
if (simmethod == "SELF")//自分でつくったサーバーに http 通信してそこに置いてある cgi から結果をもらうとき
{
URL = hostName + "/toolbox.cgi";
parameter = "org=" + HttpUtility.UrlEncode(org) +
"&back=" + HttpUtility.UrlEncode(back)+
"&callmethod=similar";
}
else if (simmethod == "NICT")
{
```

```
URL = "http://falcon.kuis.kyoto-u.ac.jp/~myzk/cgi-bin/compare2.cgi";
parameter = "org=" + HttpUtility.UrlEncode(org) +
"&back=" + HttpUtility.UrlEncode(back) +
"&callmethod=similar";
}
else if (simmethod == "BLEU")
{
URL = "http://www.lab7.kuis.kyoto-u.ac.jp/~hysd/cgi-bin/bleu/scoreonly.cgi";
parameter = "org=" + HttpUtility.UrlEncode(org) +
"&back=" + HttpUtility.UrlEncode(back) +
"&score=bleu&optx=no";
}
else if (simmethod == "NIST")
{
URL = "http://www.lab7.kuis.kyoto-u.ac.jp/~hysd/cgi-bin/bleu/scoreonly.cgi";
parameter = "org=" + HttpUtility.UrlEncode(org) +
"&back=" + HttpUtility.UrlEncode(back) +
"&score=nist&optx=no";
}
else if (simmethod == "CS")
{
URL = "http://ice.kuis.kyoto-u.ac.jp/~hysd/cgi-bin/nl/hello.cgi";
parameter = "org=" + HttpUtility.UrlEncode(org) +
"&back=" + HttpUtility.UrlEncode(back) +
"&callmethod=similar";
}


HttpWebRequest req = (HttpWebRequest)HttpWebRequest.Create(URL);
req.Proxy = GlobalProxySelection.Select;
req.Method = "POST";
req.ContentLength = parameter.Length;
req.ContentType = "application/x-www-form-urlencoded";

StreamWriter output = null;
try
{
output = new StreamWriter(req.GetRequestStream());
output.Write(parameter);//parameter をここで送る
}
catch(Exception e)
{
return e.Message;
}
finally
{
if(output != null)
{
output.Close();
}
}

StreamReader input = null;
try
{
HttpWebResponse res = (HttpWebResponse)req.GetResponse();
input = new StreamReader(res.GetResponseStream());
string ret = input.ReadToEnd();//類似度計算の結果を受け取る.
return ret;
}
catch(Exception e)
{
return e.Message;
}
finally
{
if(input != null)
{
input.Close();
}
}
```

```
}//http version end

return "-1";
}
}
}
```

A.2.5 Log.cs

```
using System;
using System.Collections;
using System.Xml.Serialization;
using System.IO;

namespace SimTrans
{
public class LogTool
{
public LogTool()
{
}

public static void WriteLog(LogMain log, string filename)
{
if(log != null)
{
XmlSerializer serializer = new XmlSerializer(typeof(LogMain));
TextWriter writer = new StreamWriter(filename);

serializer.Serialize(writer, log);
writer.Close();
}
}

public static LogMain ReadLog(string filename)
{
XmlSerializer serializer = new XmlSerializer(typeof(LogMain));
TextReader reader = new StreamReader(filename);

LogMain log = (LogMain)serializer.Deserialize(reader);
reader.Close();
return log;
}
}
public class SntfragpathData
{
public DateTime dateTime;
public double similarity;
public string source;
public string mid;
public string target;

public SntfragpathData()
{
dateTime = DateTime.Now;
similarity = 0;
source = "にほんご";
mid = "媒介言語";
target = "English";
}

public SntfragpathData(double similarity, string source, string mid, string target) :this()
{
this.similarity = similarity;
this.source = source;
this.mid = mid;
this.target = target;
}
}
```

```
public class SntfragData
{
public DateTime dateTime;
public string fragmnt;
public int score;
public double score2;

public SntfragData()
{
dateTime = DateTime.Now;
this.fragmnt = "断片";
this.score = 0;
}

public SntfragData(string fragmnt, int score) : this()
{
this.fragmnt = fragmnt;
this.score = score;
}
public SntfragData(string fragmnt, double score2) : this()
{
this.fragmnt = fragmnt;
this.score2 = score2;
}
}


public class GoodTreeData  : IComparable
{
public string sentence;
public double point;
public Similar self_sim6;
public Similar org_sim6;
public Similar try_sim6;

public string [] problem;
public string  first;
public string  second;
public string  third;

        public string simtype;

public GoodTreeData()
{
this.sentence = "hoge";
this.point = -1;

this.self_sim6 = new Similar();
this.org_sim6 = new Similar();
this.try_sim6 = new Similar();

this.problem = new string [] {};
this.first = "";
this.second = "";
this.third = "";
this.simtype = "";
}
public GoodTreeData(string sentence, Similar org_sim6, Similar try_sim6):this()
{
this.sentence = sentence;
this.org_sim6 = org_sim6.makeCopy();
this.try_sim6 = try_sim6.makeCopy();
}

public GoodTreeData(string sentence,Similar self_sim6,
 Similar org_sim6, Similar try_sim6, string [] problem,
string first, string second, string third,string simtype):this()
{
this.sentence = sentence;
this.self_sim6 = self_sim6.makeCopy();
this.org_sim6 = org_sim6.makeCopy();
```

```
this.try_sim6 = try_sim6.makeCopy();
this.problem = (string [])problem.Clone();
this.first = first;
this.second = second;
this.third = third;

this.simtype = simtype;
}
public string GetSentence()
{
return this.sentence;
}
public string [] GetProblem()
{
string [] ret= (string [])(this.problem.Clone());
return ret;
}
public double get_similarity(string str,string simtype)
{
double ret;
if(str == "try_sim")
{
ret = this.try_sim6.selectSimPoint(simtype);
return ret;
}
if(str == "org_sim")
{
ret = this.org_sim6.selectSimPoint(simtype);
return ret;
}
if(str == "self_sim")
{
ret = this.self_sim6.selectSimPoint(simtype);
return ret;
}
return 0;
}
public GoodTreeData makeCopy()
{
GoodTreeData ret = new GoodTreeData();

ret.sentence = this.sentence;
ret.point = this.point;
ret.self_sim6 = this.self_sim6.makeCopy();
ret.org_sim6 = this.org_sim6.makeCopy();
ret.try_sim6 = this.try_sim6.makeCopy();
ret.problem = (string [])this.problem.Clone();
ret.first = this.first;
ret.second = this.second;
ret.third = this.third;
ret.simtype = this.simtype;
return ret;
}

public int CompareTo(object obj)//obj:呼び出されたオブジェクト
{
if(obj == null)return 1;

if (obj is GoodTreeData)
{
GoodTreeData other = (GoodTreeData)obj;
double own1 = this.get_similarity(this.first,this.simtype);
double own2 = this.get_similarity(this.second,this.simtype);
double own3 = this.get_similarity(this.third,this.simtype);

double you1 = other.get_similarity(other.first,other.simtype);
double you2 = other.get_similarity(other.second,other.simtype);
double you3 = other.get_similarity(other.third,other.simtype);

if(own1 > you1)return 1;
if(own1 < you1)return -1;
```

```
if(own1 == you1)
{
if(own2 > you2)return 1;
if(own2 < you2)return -1;
if(own2 == you2)
{
if(own3 > you3)return 1;
if(own3 < you3)return -1;
if(own3 == you3)return 0;
}
}
}
throw new ArgumentException("object is not a GoodTreeData");
}
}
```

```
public class LogData
{
public DateTime dateTime;//記録された時刻
public OptionList option;//オプション設定

public string question;//課題
public string source;//原文
public string mid;//翻訳結果
public string target;//折り返し
public Similar sim6;

public string sprtmes;//翻訳サポートメッセージ
public string badparts;//翻訳に適さないと思われる文節群
public bns [] bunsetuMatch;//ID・文節・その文節に与えられた値

public WordPoint [] orgRanking;//断片文の　原文類似度ランキング
public WordPoint [] org_zeroRanking;//断片文の　1 st 原文類似度ランキング
public fragData [] fragRanking;//断片ごとの類似度ランク
public GoodTreeData [] goodTree;
public WordPoint [] diffRanking;
public WordPoint [] bunsetuRanking;//文節ごとの類似度ランク


public LogData()
{
this.dateTime = DateTime.Now;
this.badparts = "問題のある文節たち";//ID の集合
this.question = "課題";
this.source = "原文文";
this.mid = "入力の訳";
this.target = "折り返し翻訳結果";
this.sim6 = new Similar();
this.sprtmes = "翻訳サポートメッセージ";
}

public LogData(Similar sim6,string question, string source, string mid, string target,
double clrsimpoint, double weightpoint, double fragclrpoint, string simmethod,
string hostname, int hostport, string first_key, string second_key, string third_key,string simtype, bool mode)
{
this.option = new OptionList();

dateTime = DateTime.Now;

this.option.clrsimpoint = clrsimpoint;
this.option.weightpoint = weightpoint;
this.option.fragclrpoint = fragclrpoint;
this.option.simmethod = simmethod;
this.option.hostname = hostname;
this.option.hostport = hostport;
this.option.first_key = first_key;
this.option.second_key = second_key;
```

```
this.option.third_key = third_key;
this.option.simtype = simtype;
if(mode == true)this.option.debugmode = "ON";
else this.option.debugmode = "OFF";

this.question = question;
this.source = source;
this.mid = mid;
this.target = target;

            this.sim6 = sim6.makeCopy();

this.bunsetuMatch = new bns [] {};

this.orgRanking = new WordPoint [] {};
this.org_zeroRanking = new WordPoint [] {};
this.fragRanking = new fragData [] {};
this.bunsetuRanking = new WordPoint [] {};
this.goodTree = new GoodTreeData [] {};
this.diffRanking = new WordPoint [] {};
}
}

public class LogsntfragCollection : System.Collections.IList
{
public ArrayList sntFragList;

public LogsntfragCollection()
{
sntFragList = new ArrayList();
}

#region implementation of ICollection
public void CopyTo(System.Array array, int index)
{
sntFragList.CopyTo(array, index);
}
public int Count
{
get{return sntFragList.Count;}
}
public bool IsSynchronized
{
get{return sntFragList.IsSynchronized;}
}
public object SyncRoot
{
get{return sntFragList.SyncRoot;}
}
#endregion
#region implementation of IEnumerable
public IEnumerator GetEnumerator()
{
return sntFragList.GetEnumerator();
}
#endregion
#region implementation of IList
public int Add(object value)
{
return sntFragList.Add(value);
}
public void Clear()
{
sntFragList.Clear();
}
public bool Contains(object item)
{
return sntFragList.Contains(item);
}
public int IndexOf(object value)
{
```

```
return sntFragList.IndexOf(value);
}
public void Insert(int index, object value)
{
sntFragList.Insert(index, value);
}
public bool IsFixedSize
{
get{return sntFragList.IsFixedSize;}
}
public bool IsReadOnly
{
get{return sntFragList.IsReadOnly;}
}
public void Remove(object obj)
{
sntFragList.Remove(obj);
}
public void RemoveAt(int index)
{
sntFragList.RemoveAt(index);
}
object IList.this[int index]
{
get{return sntFragList[index];}
set{sntFragList[index] = value;}
}
public SntfragData this[int index]
{
get{return (SntfragData)sntFragList[index];}
set{sntFragList[index] = value;}
}
#endregion

}

public class LogDataCollection : System.Collections.IList
{
public ArrayList logDataList;

public LogDataCollection()
{
logDataList = new ArrayList();
}

#region implementation of ICollection
public void CopyTo(System.Array array, int index)
{
logDataList.CopyTo(array, index);
}
public int Count
{
get{return logDataList.Count;}
}
public bool IsSynchronized
{
get{return logDataList.IsSynchronized;}
}
public object SyncRoot
{
get{return logDataList.SyncRoot;}
}
#endregion
#region implementation of IEnumerable
public IEnumerator GetEnumerator()
{
return logDataList.GetEnumerator();
}
#endregion
#region implementation of IList
public int Add(object value)
```

```
{
return logDataList.Add(value);
}
public void Clear()
{
logDataList.Clear();
}
public bool Contains(object item)
{
return logDataList.Contains(item);
}
public int IndexOf(object value)
{
return logDataList.IndexOf(value);
}
public void Insert(int index, object value)
{
logDataList.Insert(index, value);
}
public bool IsFixedSize
{
get{return logDataList.IsFixedSize;}
}
public bool IsReadOnly
{
get{return logDataList.IsReadOnly;}
}
public void Remove(object obj)
{
logDataList.Remove(obj);
}
public void RemoveAt(int index)
{
logDataList.RemoveAt(index);
}
object IList.this[int index]
{
get{return logDataList[index];}
set{logDataList[index] = value;}
}
public LogData this[int index]
{
get{return (LogData)logDataList[index];}
set{logDataList[index] = value;}
}
#endregion

}

public class LogsntfragpathCollection : System.Collections.IList
{
public ArrayList sntfragPathList;

public LogsntfragpathCollection()
{
sntfragPathList = new ArrayList();
}

#region implementation of ICollection
public void CopyTo(System.Array array, int index)
{
sntfragPathList.CopyTo(array, index);
}
public int Count
{
get{return sntfragPathList.Count;}
}
public bool IsSynchronized
{
get{return sntfragPathList.IsSynchronized;}
}
```

```
public object SyncRoot
{
get{return sntfragPathList.SyncRoot;}
}
#endregion
#region implementation of IEnumerable
public IEnumerator GetEnumerator()
{
return sntfragPathList.GetEnumerator();
}
#endregion
#region implementation of IList
public int Add(object value)
{
return sntfragPathList.Add(value);
}
public void Clear()
{
sntfragPathList.Clear();
}
public bool Contains(object item)
{
return sntfragPathList.Contains(item);
}
public int IndexOf(object value)
{
return sntfragPathList.IndexOf(value);
}
public void Insert(int index, object value)
{
sntfragPathList.Insert(index, value);
}
public bool IsFixedSize
{
get{return sntfragPathList.IsFixedSize;}
}
public bool IsReadOnly
{
get{return sntfragPathList.IsReadOnly;}
}
public void Remove(object obj)
{
sntfragPathList.Remove(obj);
}
public void RemoveAt(int index)
{
sntfragPathList.RemoveAt(index);
}
object IList.this[int index]
{
get{return sntfragPathList[index];}
set{sntfragPathList[index] = value;}
}
public SntfragpathData this[int index]
{
get{return (SntfragpathData)sntfragPathList[index];}
set{sntfragPathList[index] = value;}
}
#endregion

}

public class OptionList
{
public double clrsimpoint;
public double weightpoint;
public double fragclrpoint;
public string simmethod;
public string hostname;
public int hostport;
public string first_key;
```

```
public string second_key;
public string third_key;
public string simtype;//recall precision f-value...

public string debugmode;

public OptionList()
{
this.clrsimpoint = 0;
this.weightpoint = 0;
this.fragclrpoint = 0;
this.simmethod = "nothing";
this.hostname = "";
this.hostport = 8080;
this.debugmode = "OFF";
this.simtype = "";
}
}

public class WordPointList : System.Collections.IList
{
public ArrayList WPList;
public WordPointList()
{
this.WPList = new ArrayList();
}
#region implementation of ICollection
public void CopyTo(System.Array array, int index)
{
WPList.CopyTo(array, index);
}
public int Count
{
get{return WPList.Count;}
}
public bool IsSynchronized
{
get{return WPList.IsSynchronized;}
}
public object SyncRoot
{
get{return WPList.SyncRoot;}
}
#endregion
#region implementation of IEnumerable
public IEnumerator GetEnumerator()
{
return WPList.GetEnumerator();
}
#endregion
#region implementation of IList
public int Add(object value)
{
return WPList.Add(value);
}
public void Clear()
{
WPList.Clear();
}
public bool Contains(object item)
{
return WPList.Contains(item);
}
public int IndexOf(object value)
{
return WPList.IndexOf(value);
}
public void Insert(int index, object value)
{
WPList.Insert(index, value);
}
```

```
public bool IsFixedSize
{
get{return WPList.IsFixedSize;}
}
public bool IsReadOnly
{
get{return WPList.IsReadOnly;}
}
public void Remove(object obj)
{
WPList.Remove(obj);
}
public void RemoveAt(int index)
{
WPList.RemoveAt(index);
}
object IList.this[int index]
{
get{return WPList[index];}
set{WPList[index] = value;}
}
public SntfragpathData this[int index]
{
get{return (SntfragpathData)WPList[index];}
set{WPList[index] = value;}
}
#endregion
}

public class LogResult
{
public int number;
public LogDataCollection logDataList;
public LogQuestionCollection logQuestionList;

public LogResult()
{
number = 1;
logDataList = new LogDataCollection();
logQuestionList = null;
}

public LogResult(int n) : this()
{
number = n;
}
}

public class LogResultCollection : System.Collections.IList
{
public ArrayList logResultList;

public LogResultCollection()
{
logResultList = new ArrayList();
}

#region implementation of ICollection
public void CopyTo(System.Array array, int index)
{
logResultList.CopyTo(array, index);
}
public int Count
{
get{return logResultList.Count;}
}
public bool IsSynchronized
{
get{return logResultList.IsSynchronized;}
}
public object SyncRoot
```

```
{
get{return logResultList.SyncRoot;}
}
#endregion
#region implementation of IEnumerable
public IEnumerator GetEnumerator()
{
return logResultList.GetEnumerator();
}
#endregion
#region implementation of IList
public int Add(object value)
{
return logResultList.Add(value);
}
public void Clear()
{
logResultList.Clear();
}
public bool Contains(object item)
{
return logResultList.Contains(item);
}
public int IndexOf(object value)
{
return logResultList.IndexOf(value);
}
public void Insert(int index, object value)
{
logResultList.Insert(index, value);
}
public bool IsFixedSize
{
get{return logResultList.IsFixedSize;}
}
public bool IsReadOnly
{
get{return logResultList.IsReadOnly;}
}
public void Remove(object obj)
{
logResultList.Remove(obj);
}
public void RemoveAt(int index)
{
logResultList.RemoveAt(index);
}
object IList.this[int index]
{
get{return logResultList[index];}
set{logResultList[index] = value;}
}
public LogResult this[int index]
{
get{return (LogResult)logResultList[index];}
set{logResultList[index] = value;}
}
#endregion

}

public class LogUserData
{
public string name;
public LogQuestionCollection userQuestionList;

public LogUserData()
{
name = "hoge";
}
```

```
public LogUserData(string name)
{
this.name = name;
}
}

public enum QuestionType
{
Radio,
Check,
Free
}

public class LogQuestion
{
public QuestionType type;
public string question;
public string answer;

public LogQuestion()
{
type = QuestionType.Free;
question = "q";
answer = "a";
}

public LogQuestion(QuestionType t, string q, string a)
{
type = t;
question = q;
answer = a;
}
}

public class LogQuestionCollection : System.Collections.IList
{
public ArrayList logQuestionList;

public LogQuestionCollection()
{
logQuestionList = new ArrayList();
}

#region implementation of ICollection
public void CopyTo(System.Array array, int index)
{
logQuestionList.CopyTo(array, index);
}
public int Count
{
get{return logQuestionList.Count;}
}
public bool IsSynchronized
{
get{return logQuestionList.IsSynchronized;}
}
public object SyncRoot
{
get{return logQuestionList.SyncRoot;}
}
#endregion
#region implementation of IEnumerable
public IEnumerator GetEnumerator()
{
return logQuestionList.GetEnumerator();
}
#endregion
#region implementation of IList
public int Add(object value)
{
return logQuestionList.Add(value);
```

```
}
public void Clear()
{
logQuestionList.Clear();
}
public bool Contains(object item)
{
return logQuestionList.Contains(item);
}
public int IndexOf(object value)
{
return logQuestionList.IndexOf(value);
}
public void Insert(int index, object value)
{
logQuestionList.Insert(index, value);
}
public bool IsFixedSize
{
get{return logQuestionList.IsFixedSize;}
}
public bool IsReadOnly
{
get{return logQuestionList.IsReadOnly;}
}
public void Remove(object obj)
{
logQuestionList.Remove(obj);
}
public void RemoveAt(int index)
{
logQuestionList.RemoveAt(index);
}
object IList.this[int index]
{
get{return logQuestionList[index];}
set{logQuestionList[index] = value;}
}
public LogQuestion this[int index]
{
get{return (LogQuestion)logQuestionList[index];}
set{logQuestionList[index] = value;}
}
#endregion
}

public class LogMain
{
public LogUserData person;
public LogQuestionCollection preQuestionList;
public LogResultCollection logResultList;
public LogQuestionCollection postQuestionList;

public LogMain()
{
person = null;
logResultList = new LogResultCollection();
}

public LogMain(string name) : this()
{
person = new LogUserData(name);
}

}


}
```

A.2.6 HighlightRichTextBox.cs

```
using System;
```

```
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.IO;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading;

namespace SimTrans
{
public class HighlightRichTextBox : System.Windows.Forms.RichTextBox
{
public void BackColorSetWhole(Color backColorDefault)
{
this.SelectAll();
this.SelectionBackColor = backColorDefault;
}
public Color SelectionBackColor
{
set
{
if(this.SelectedText == null)
return;
if(this.SelectedText == "")
return;
StringBuilder sb = new StringBuilder(); //use StringBuilder for speed and cleanliness
string selText = this.SelectedRtf; //move to local string for speed
string strTemp = null; //used twice for ease of calculating internal coordinates
int fontTableEnds; //end character of the rtf font table
int colorTableBegins; //beginning of the rtf color table
int colorTableEnds; //end of the rtf color table
int startLooking; //used to walk a string appending chunks
int highlightBlockStart; //used to find "\highlight#" block for stripping
int highlightBlockEnd; //used to find "\highlight#" block for stripping
int newColorIndex = 0; //new color table index for incoming color

fontTableEnds = selText.IndexOf("}}",0);
//add the header and font table to the string accumulator
sb.Append(selText.Substring(0, fontTableEnds+2));
//find the color table start
colorTableBegins = selText.IndexOf("{\\colortbl", fontTableEnds);
if(colorTableBegins == -1)//no color table exists
{
//add a color table header
sb.Append("{\\colortbl ;");
//no color table so for later use make the ColorTableEnd the same as FontTableEnds
colorTableEnds = fontTableEnds;
//default our new color table index to 1 since it will be the only one
//remember Color table index 0 is reserved
newColorIndex = 1;
}
else //a color table already exists
{
//find the end of the color table
colorTableEnds = selText.IndexOf("}", colorTableBegins);
//backup one character so as to exclude the brace
colorTableEnds -= 1;
//need to count the quantity of semi;colons which will
//... determine what color table index number our new color will be
strTemp = selText.Substring(fontTableEnds + 2, (colorTableEnds-fontTableEnds) - 1);
for(int i=1; i < strTemp.Length; i++)
{
if(strTemp.Substring(i, 1) == ";")
{
newColorIndex += 1;
}
}
//append the color table without end brace
sb.Append(strTemp);
```

```
}//end if
//append the color table entry for the highlight color
sb.Append("\\red" + value.R.ToString().Trim());
sb.Append("\\green" + value.G.ToString().Trim());
sb.Append("\\blue" + value.B.ToString().Trim());
//append the table entry terminator semi;colon
sb.Append(";");
//append the color table terminating brace
sb.Append("}");
//append the new highlight tag
sb.Append("\\highlight" + newColorIndex.ToString().Trim());
//Drop into a single string for easier manipulation
strTemp = selText.Substring(colorTableEnds+2, (selText.Length - colorTableEnds) - 2);
//begin at first character
startLooking = 0;
//append everything remaining, but strip all remaining highlight tags
while(true)
{
//find a "\highlight" block
highlightBlockStart = strTemp.IndexOf("\\highlight", startLooking);
//if no more "\highlight" block found
if(highlightBlockStart == -1)
{
//append everything remaining
sb.Append(strTemp.Substring(startLooking, strTemp.Length-startLooking));
//we're done appending
break;
}
//calculate the end of the word "highlight"
highlightBlockEnd = highlightBlockStart + 9;
//accomodate color tables with over 9 colors and thus multi-digit color indexes
//Plus, watch for (and discard) ONE space if it immediately follows the last digit
do
{

//keep stepping past end
highlightBlockEnd += 1;
//watch for (and discard) ONE space if it immediately follows the last digit
if(strTemp.Substring(highlightBlockEnd+1, 1) == " ")
{

highlightBlockEnd += 1;
break;

}
//looking for the first non-numeric character
}while("0123456789".IndexOf(strTemp.Substring(highlightBlockEnd + 1, 1), 1) > 0);
//append this block
sb.Append(strTemp.Substring(startLooking, (highlightBlockStart-startLooking)));
//move the start forward past the last "\highlight#" block
startLooking = highlightBlockEnd + 1;
}
this.SelectedRtf = sb.ToString();
}//end set
}//end function
}

public class hoge{
int huga;
}
}
```

A.2.7 bns.cs

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.IO;
using System.Text;
```

```
using System.Text.RegularExpressions;
using System.Threading;

namespace SimTrans
{
public class HighlightRichTextBox : System.Windows.Forms.RichTextBox
{
public void BackColorSetWhole(Color backColorDefault)
{
this.SelectAll();
this.SelectionBackColor = backColorDefault;
}
public Color SelectionBackColor
{
set
{
if(this.SelectedText == null)
return;
if(this.SelectedText == "")
return;
StringBuilder sb = new StringBuilder(); //use StringBuilder for speed and cleanliness
string selText = this.SelectedRtf; //move to local string for speed
string strTemp = null; //used twice for ease of calculating internal coordinates
int fontTableEnds; //end character of the rtf font table
int colorTableBegins; //beginning of the rtf color table
int colorTableEnds; //end of the rtf color table
int startLooking; //used to walk a string appending chunks
int highlightBlockStart; //used to find "\highlight#" block for stripping
int highlightBlockEnd; //used to find "\highlight#" block for stripping
int newColorIndex = 0; //new color table index for incoming color

fontTableEnds = selText.IndexOf("}}",0);
//add the header and font table to the string accumulator
sb.Append(selText.Substring(0, fontTableEnds+2));
//find the color table start
colorTableBegins = selText.IndexOf("{\\colortbl", fontTableEnds);
if(colorTableBegins == -1)//no color table exists
{
//add a color table header
sb.Append("{\\colortbl ;");
//no color table so for later use make the ColorTableEnd the same as FontTableEnds
colorTableEnds = fontTableEnds;
//default our new color table index to 1 since it will be the only one
//remember Color table index 0 is reserved
newColorIndex = 1;
}
else //a color table already exists
{
//find the end of the color table
colorTableEnds = selText.IndexOf("}", colorTableBegins);
//backup one character so as to exclude the brace
colorTableEnds -= 1;
//need to count the quantity of semi;colons which will
//... determine what color table index number our new color will be
strTemp = selText.Substring(fontTableEnds + 2, (colorTableEnds-fontTableEnds) - 1);
for(int i=1; i < strTemp.Length; i++)
{
if(strTemp.Substring(i, 1) == ";")
{
newColorIndex += 1;
}
}
//append the color table without end brace
sb.Append(strTemp);
}//end if
//append the color table entry for the highlight color
sb.Append("\\red" + value.R.ToString().Trim());
sb.Append("\\green" + value.G.ToString().Trim());
sb.Append("\\blue" + value.B.ToString().Trim());
//append the table entry terminator semi;colon
sb.Append(";");
```

```
//append the color table terminating brace
sb.Append("}");
//append the new highlight tag
sb.Append("\\highlight" + newColorIndex.ToString().Trim());
//Drop into a single string for easier manipulation
strTemp = selText.Substring(colorTableEnds+2, (selText.Length - colorTableEnds) - 2);
//begin at first character
startLooking = 0;
//append everything remaining, but strip all remaining highlight tags
while(true)
{
//find a "\highlight" block
highlightBlockStart = strTemp.IndexOf("\\highlight", startLooking);
//if no more "\highlight" block found
if(highlightBlockStart == -1)
{
//append everything remaining
sb.Append(strTemp.Substring(startLooking, strTemp.Length-startLooking));
//we're done appending
break;
}
//calculate the end of the word "highlight"
highlightBlockEnd = highlightBlockStart + 9;
//accomodate color tables with over 9 colors and thus multi-digit color indexes
//Plus, watch for (and discard) ONE space if it immediately follows the last digit
do
{

//keep stepping past end
highlightBlockEnd += 1;
//watch for (and discard) ONE space if it immediately follows the last digit
if(strTemp.Substring(highlightBlockEnd+1, 1) == " ")
{

highlightBlockEnd += 1;
break;

}
//looking for the first non-numeric character
}while("0123456789".IndexOf(strTemp.Substring(highlightBlockEnd + 1, 1), 1) > 0);
//append this block
sb.Append(strTemp.Substring(startLooking, (highlightBlockStart-startLooking)));
//move the start forward past the last "\highlight#" block
startLooking = highlightBlockEnd + 1;
}
this.SelectedRtf = sb.ToString();
}//end set
}//end function
}

public class hoge{
int huga;
}
}
```