特別研究報告書

協調翻訳の修正履歴を用いた
コミュニティ辞書の開発

指導教員　　石田 亨 教授

京都大学工学部情報学科

山本章平

平成27年1月30日

協調翻訳の修正履歴を用いたコミュニティ辞書の開発

山本章平

内容梗概

近年，Twitter や Facebook などの SNS は世界中の人々が利用しており，自由に交流が可能である．しかし，実際には言語の壁が自由なコミュニケーションを妨げている．機械翻訳は言語の壁を取り除くための有望なツールであるが．その翻訳の精度は人手による翻訳に比べて低く，特に新語や専門用語に対する誤訳の可能性は高い．

機械翻訳の精度を人手により補う手法として協調翻訳というものが存在する．協調翻訳は複数の作業者が協力して翻訳を行なう手法であり，翻訳専門家でなくとも．互いに長所を生かすことで良い翻訳を得る可能性が高くなる．協調翻訳は人手による翻訳のため，SNS のコミュニティ特有の単語や新語であっても翻訳できる可能性があり，SNS 上の文章の翻訳が必要となった場合にコミュニティのメンバーによる協調翻訳を行なうことが考えられる．

しかし，協調翻訳を SNS で行なう手法は存在しない．さらに，SNS コミュニティの文章の翻訳を協調翻訳で行なったとしても，その結果は再利用されずコミュニティでの機械翻訳の精度向上には生かされない．この場合，機械翻訳で翻訳不可能な単語を含む文章の翻訳を何度も協調翻訳で行なわなければならなくなる．

本研究の目的は，協調翻訳結果をコミュニティでの機械翻訳の精度改善に利用することである．本研究で取り組む課題は以下の２つである．

1. SNS における協調翻訳手法の設計
   現在 SNS 上で協調翻訳を行なう手法は存在しないため，SNS 上で協調翻訳を行なうための手法を設計する必要がある．
2. 協調翻訳結果を用いてコミュニティでの機械翻訳精度を改善するシステムの設計と実装
   現在協調翻訳は機械翻訳の精度改善に利用されておらず，同じ文章を何度も協調翻訳で翻訳しなければならないという状況が生じる可能性がある．SNS コミュニティでの協調翻訳結果を利用して，コミュニティ内での機械翻訳の精度を向上させることのできるシステムが必要である．

上記の課題を解決するために，本研究ではSNS上での協調翻訳手法を設計した．さらに，SNS上の協調翻訳結果から辞書を生成し，その辞書を用いた機械翻訳を利用することで，コミュニティでの機械翻訳の精度を向上させるシステムを実装した．

本研究での協調翻訳は，原文を機械翻訳で翻訳し，修正の必要がある場合に人手による翻訳を行なうという手法を用いる．この手法をもとに，原文と機械翻訳を同時投稿し，修正を返信として投稿するというSNSで利用可能な手法を設計した．

つぎに，上記手法を利用しTwitter上で協調翻訳を行なうことが可能なクライアントアプリケーションを実装した．その後，クライアントアプリケーションでの協調翻訳結果から修正履歴をたどることで，コミュニティ辞書を構築し，その辞書を利用した機械翻訳を行なうことのできるサーバーアプリケーションを実装した．

本研究の貢献は以下の２つである．

1. SNSにおける協調翻訳手法の設計
   SNS上で協調翻訳を行なう手法として原文と機械翻訳文を同時に投稿し，機械翻訳文に修正が必要な場合に返信機能を使って修正を行なうという手法を提案した．
2. 協調翻訳の修正結果を用いたコミュニティ辞書生成システムの設計と実装
   協調翻訳をTwitter上で行なうことのできるクライアントアプリケーションを実装し，その協調翻訳結果から修正履歴をたどることでコミュニティ辞書を生成するサーバーアプリケーションを実装した．

Developing Community Dictionaries Using Editing History in Collaborative Translation

Shohei Yamamoto

Abstract

Nowadays, people all over the world communicate with each other freely using the SNS like Twitter and Facebook. However, language barrier is preventing the free communication in reality. Machine translation(MT) is a promising tool for breaking the language barrier, but the accuracy of machine translation is lower than human translation especially for new words and technical terms.

Collaborative translation is a way to compensate for the MT accuracy. Collaborative translation is a translation technique that multiple participants cooperate to translate and improve translation accuracy and fluency. It is possible to get good translation results without experts by collaborative translation. Because collaborative translation is manual, special terms in community and new words can be translated. Therefore, when texts in a SNS community need to be translated, community members can translate them by collaborative translation.

However, the method of collaborative translation in SNS does not exist now. Moreover, even though a text in SNS community can be translated by collaborative translation, the result of the translation doesn't reflect to the MT accuracy. Sentences that include words and expressions peculiar to communities are repeatedly used in the same community, thus collaborative translation must be performed over and over again to get the same translation.

The purpose of this study is reusing the collaborative translation result for improving MT accuracy. There are two problems to be solved.

1. **The design of the method to perform collaborative translation in SNS**

   The method of collaborative translation in SNS does not exist now. There is a need to design such a method.

2. **The design and implementation of the system for improving MT accuracy of community terms by using collaborative translation results**

Collaborative translation doesn't affect the accuracy of MT now. It is necessary to design and implement the system to improve MT accuracy of community terms by using collaborative translation results.

To solve the problems, I designed the method of collaborative translation in SNS. And I implemented the system that generates community dictionaries from collaborative translation results. The dictionaries are combined with MT and improve MT accuracy.

The method of collaborative translation in this study is as follows. First, the original text is translated by MT. Second, if the translation needs to be corrected , it is modified manually. The modifications continue until getting good translation. Based on this method, I designed a method available in SNS. Original text and its translation by MT are posted simultaneously and community members modify the translation using the reply function.

Next, I implemented client application of the above method, which is used for collaborative translation in Twitter. And I implemented server application that can generate bilingual dictionary of community words by tracing the histories of collaborative translation results of client application. Client application calls MT with the dictionaries from server application, the collaborative translation results are reused for MT.

The contributions of this study are as follows.

1. **The design of the method to perform collaborative translation in SNS**

   I designed the method of collaborative translation in SNS. Original text and its translation by MT are posted simultaneously and community members modify the translation using the reply function.

2. **The design and implementation of the system that generates a community dictionaries from collaborative translation results**

   I implemented client application of the above method, which is used for collaborative translation in Twitter. And I implemented server application that can generate bilingual dictionaries of community words by tracing the histories of collaborative translation results of client application.

協調翻訳の修正履歴を用いたコミュニティ辞書の開発

目次

第 1 章　はじめに　1

第 2 章　関連研究　3

2.1　言語グリッド 3

2.2　対訳辞書 3

2.2.1　対訳辞書を用いた機械翻訳 3

2.2.2　対訳辞書の生成手法 5

2.3　協調翻訳 5

第 3 章　コミュニティにおける協調翻訳　7

3.1　SNS 協調翻訳 7

3.2　協調翻訳からのコミュニティ辞書構築 9

第 4 章　協調翻訳からの対訳抽出　9

第 5 章　Twitter 協調翻訳支援システム　13

5.1　システムの概要 14

5.2　クライアントアプリのユーザーインターフェイスと実行例 15

5.2.1　ユースケース 15

5.2.2　ユーザーインターフェイス 15

5.2.3　プログラムの構成 17

5.3　サーバーの実装 17

第 6 章　実験と考察　20

6.1　実験手法と目的 20

6.2　実験結果と考察 21

第 7 章　おわりに　22

謝辞　23

参考文献　23

付録：主なソースコード　A-1

A.1　クライアントアプリケーション A-1

A.1.1 CollaborativeTranslation.java A-1
A.1.2 OAuthActivity.java A-2
A.1.3 LanguageGrid.java A-4
A.1.4 ExpandedStatus.java A-6
A.1.5 Twitter.java A-7
A.1.6 tweet.html A-12
A.1.7 tweet.js A-15
A.2 サーバーアプリケーション A-24
A.2.1 TweetStore.java A-24
A.2.2 Servlet.java A-25
A.2.3 EnEnAlignmentCreator.java A-26
A.2.4 JaEnAlignmentCreator.java A-27
A.2.5 History.java A-29
A.2.6 MatchingStore.java A-30
A.2.7 BDictServlet.java A-36
A.2.8 CompositeServiceServlet.java A-37

第1章　はじめに

インターネット技術の進歩や，インターネットにアクセス可能な機器の普及により，インターネットの利用者は現在も増加し続けている．インターネットは世界中の様々な言語や文化，知識を持った人々が利用しており，また，その人々が発信する様々な情報にアクセス可能である．しかし，技術的にはアクセス可能であるが，インターネット上の情報は様々な言語で記述されており，言語の壁により実際に理解できる情報は限られる．

機械翻訳はある言語で記述された文章を他の言語の文章に翻訳する技術である．機械翻訳は言語の壁を取り除くための技術として期待されており．全く理解ができない文章でも，機械翻訳によって理解できる言語に翻訳すれば，その文章の情報を理解することが可能である．しかし，機械翻訳の翻訳の精度は人手に比べて低く，実用可能な品質の翻訳を常に得ることはできない．特に，特定のコミュニティのみで使われる用語や専門用語などに対しての翻訳の精度は著しく低い．

この問題に対して，本研究では対訳辞書を用いた機械翻訳を利用する．対訳辞書を用いた機械翻訳は機械翻訳の精度を向上させるための方法の一つであり，ある用語に対しての翻訳元の言語と翻訳後の言語の対訳をあらかじめ登録しておくことで，機械翻訳を行なう際にその対訳辞書に含まれる対訳を優先的に利用する．これによって，ある専門分野やコミュニティに関する用語を対訳辞書に登録しておけば，対訳辞書を用いた機械翻訳により品質の高い翻訳を得ることができるようになる．

対訳辞書を生成する手法には手作業による手法や，対訳文書や対訳コーパスを利用して自動抽出する方法 [8] が存在する．対訳辞書の手作業での生成はコストがかかり，さらに対訳文書や対訳コーパスを利用して自動抽出する方法に関しても，その対訳文書自体が手作業で生成されることが多く，コミュニティの対訳コーパスなどあるコミュニティのみにしか利用価値の無いものに関しては生成されること自体が少ない．

これに対して，複数の作業者が協力して翻訳を行なう協調翻訳という手法が存在する [1][2]．単純に非専門家の翻訳家が翻訳を行なうだけではその品質は保証されないが，複数人で協力し，他の作業者の翻訳を参照しながら翻訳を行なうことにより，それぞれの短所を補い合い，良い品質の翻訳を得ることが可能

である．

しかし，協調翻訳を行なうためのツールは既に存在しているが，協調翻訳の作業結果を言語資源として再利用する仕組みというものは存在しないという問題がある．たとえば，同じコミュニティではそのコミュニティ特有の単語や表現を含んだ会話が行なわれる可能性が高いが，その文章に対する翻訳結果が機械翻訳などに反映されないために，再び同様の文章の翻訳が必要となった場合に，再度協調翻訳作業を行なって翻訳を得なければならなくなる．協調翻訳は複数の作業者による翻訳が行なわれるため，機械翻訳に登録されていない新語や専門分野の単語についても翻訳できる可能性があり，その再利用を行なうことは大変重要である．

つまり，協調翻訳の結果を対訳辞書に登録しておけば，その対訳辞書を利用した機械翻訳を利用することで，一度協調翻訳によって翻訳された文章の翻訳を再度協調翻訳をするといった非効率的な行為は行なわれなくなる．さらに，機械翻訳によって正確に翻訳ができなかった単語の修正を協調翻訳から抽出し，対訳辞書に登録しておけば，文章だけではなくその単語の出る全ての文章に利用できる．

本研究では協調翻訳結果から対訳を抽出しその対訳を対訳辞書に登録していくことで，協調翻訳結果の再利用を行なう．その実現のためには次の 2 つの課題が生じる．

1. SNS における協調翻訳手法の設計
   現在 協調翻訳はさまざまな場面で行なわれているが，SNS 上で協調翻訳を行なう手法は確立されておらず，SNS 上で協調翻訳を行なうための手法を設計する必要がある．
2. 抽出した対訳を機械翻訳に利用するシステムの設計と実装
   協調翻訳結果から対訳を抽出するだけではなく, 対訳を辞書に登録し, その対訳辞書を利用した機械翻訳を実際にユーザーが利用できるようなシステムが必要である．

本稿では第 2 章で本研究に利用される言語グリッド，対訳辞書を用いた機械翻訳，協調翻訳などの関連研究について紹介し，第 3 章で本研究の手法やその実装を行なう際に想定したシナリオについて説明を行なう．第 4 章では協調翻訳の結果から対訳を抽出する手法について提案し，第 5 章では第 4 章の手法を用いて実装したシステムについて説明する．最後に第 6 章で本研究における成

果や今後の課題についてまとめる．

第2章　関連研究

本章では，本研究に利用している言語グリッドや対訳辞書を用いた機械翻訳について述べる．

2.1　言語グリッド

言語グリッド (Language Grid)[1)][3] は，機械翻訳などの言語資源を共有可能にするための多言語サービス基盤である．近年，グローバリゼーションにより多言語コミュニケーションに対する需要が急速に高まり，それに比例して言語資源が急激に増加しているが，知的保護の問題や機能の違いにより利用可能性に優れない．その問題に対して，言語グリッドは言語資源のユーザビリティやアクセシビリティを向上させるため，言語資源を標準の Web サービスの形として提供し，世界中のどこからでも言語サービスの利用を可能にしている．言語グリッドには，対訳翻訳 や機械翻訳などを組み合わせる「水平型言語グリッド」と，コミュニティに特化したサービスを生み出す「垂直型言語グリッド」の 2 種類の機能がある. 言語グリッドによって，インターネット上から様々な言語サービスを組み合わせて利用することが可能であり，さらにユーザーが新たに言語資源を登録することが可能になっている．言語グリッドを使って新しいツールを作る研究が複数存在し，多言語におけるチャットツールの開発などが行なわれている [4][5].

本研究の機械翻訳、形態素解析、辞書連携機械翻訳のサービスは言語グリッドのサービスによって実現されている．

2.2　対訳辞書

2.2.1　対訳辞書を用いた機械翻訳

対訳辞書とは翻訳元の単語と翻訳先の単語を登録した辞書である．機械翻訳の精度を向上させるための一つの方法として対訳辞書を用いた機械翻訳がある．対訳辞書を用いた機械翻訳では，機械翻訳に登録されている一般的な用語を登録した辞書よりも，参照する対訳辞書の用語を優先的に翻訳してから機械翻訳

[1)] http://langrid.org/jp/index.html

を行なう．よって，通常の機械翻訳では翻訳できない単語に対しても，対訳辞書に登録されていれば，精度の高い翻訳を得ることが可能である．このようなことから対訳辞書を利用した機械翻訳は，専門用語を高い精度で翻訳する必要性のあるような環境に有効であり，医療における対訳を利用した研究も行われている [6][7].

対訳辞書の例として，言語グリッドの「京大翻訳！」[1] がある.「京大翻訳！」では，京都大学内の建物や京都大学付近の道路などの名称が登録された辞書を利用しており，京都大学に関連する用語に対して高い品質の翻訳を得ることができるサービスである.

表 1: 図 1 の対訳辞書抜粋部分

日本語	英語
…	…
時計台生協ショップ	Clock Tower Co-op Shop
…	…

図 1: 通常の機械翻訳と「京大翻訳！」による結果の相違

「京大翻訳！」を使用した場合とそうでない場合の翻訳例を図 1 に示した．図 1 の文は上から，日本語の原文，通常の英語への機械翻訳結果,「京大翻訳！」による英語への機械翻訳結果となっている．また，図 1 にて用いられた対訳辞書の抜粋を表 1 に示す．図 1 から，通常の翻訳では“ 時計台生協ショップ ”は“ a clock tower livelihood cooperative association shop ”といった不自然な翻訳になっているのに対して，対訳辞書を利用している「京大翻訳！」による翻訳では“ Clock Tower Co-op Shop ”というように，京都大学内で公式に用いられて

[1] http://www.s-coop.net/smarttrans.html

いる表現となっている．

2.2.2 対訳辞書の生成手法

日英の対訳辞書の作成手法は熊野らの「対訳文書からの機械翻訳専門用語辞書作成」[8] の研究により確立された．熊野らの手法は，対訳文書の原文と翻訳文の語句の対応関係を求めるために言語的な情報と統計的な方法を利用するという手法である．この手法ではまず，ユニットと呼ばれる文，節，句などに相当する単位で日英の対応付けを行なう．その後日本語文書だけを構文解析し，語彙的構造的特徴から専門用語を抽出し，英語テキスト中の出現頻度に基づく統計情報から推定される訳語情報を抽出する．

対訳辞書生成手法としては他にも形態素解析と辞書引きによる結果から得られる統計情報を利用する手法 [9] や，単語の出現回数による閾値によって適合率を高める手法 [10]，頻度付きの共起語集合の類似度により対訳語のペアを抽出する手法 [11] などがある．

いずれの方法も，原文と翻訳文の単語の対応関係を統計情報などの手法によって同定している．もし原文と翻訳文の単語の対応の信頼度の低いものがある場合には，基本的には辞書には登録しない．SVM(Support Vector Machines) を利用して信頼度の低い訳語候補を排除し，適合率を向上させる研究 [12] も行われている．

2.3 協調翻訳

機械翻訳の精度を向上させる方法として，対訳辞書を用いた機械翻訳についての説明を行なったが，対訳辞書を用いて機械翻訳を行なったとしても，その翻訳の品質は人手による翻訳に比べて劣っている問題がある．機械翻訳の翻訳精度を人手によって補う方法として，協調翻訳という手法がある．協調翻訳は複数人の作業者によって協力して原文を翻訳するという手法である．複数の作業者が協力することで，非専門家の翻訳者であっても高い精度の翻訳結果を得ることが可能である [13]．協調翻訳には様々な方法があり，翻訳元，翻訳先それぞれの言語を母語とするモノリンガル同士で協力して翻訳文の精度を上げていく手法 [14] や，複数の作業者が他の人の作業結果を見ながら次々に翻訳を行なっていくという方法がある．この手法は繰り返し翻訳とも呼ばれ，本研究ではこの手法を利用する．協調翻訳のプロセスのフローを図 2 に示す．

本研究では，協調翻訳は次の手順で行なう．

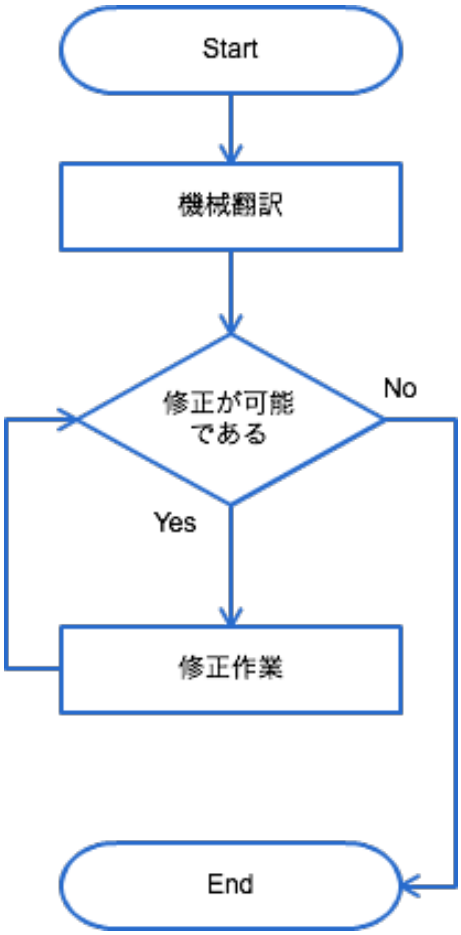

図 2: 協調翻訳の例

1. 原文を機械翻訳で翻訳する
2. 機械翻訳文が間違っていれば，作業者が修正を行なう
3. 修正文がさらに修正可能であれば他の作業者が修正をおこなう．

日本語から英語への機械翻訳に対する協調翻訳の例を図 3 に示す．

このように，原文の日本語から英語への機械翻訳をまず行ない，複数の作業者によって修正するのが協調翻訳であり，また、例にもあるように一人の作業は少なくても良く，複数の作業者がそれぞれの知識を使うことで，翻訳文全体の精度を高めることが可能である．なお、例では単語のみの修正を行なっているが、実際には文法や文の構成などの修正によって文の流暢さを向上させるということも行なわれる．

図 3: 協調翻訳の例

第 3 章　コミュニティにおける協調翻訳

3.1　SNS 協調翻訳

前章では本研究の関連研究について説明した．本章では，本研究での課題解決にあたって想定したシナリオとそのシナリオの中での協調翻訳の手法について述べる．

Twitter[1] や Facebook[2] など SNS は多くの人々が日常的に利用するツールとなっている．SNS は世界中の人々が利用でき，言語を問わず交流が可能である．現在，wikipedia[3] やマイクロソフトのマニュアル翻訳などインターネット上で協力して辞書やマニュアルなどの翻訳を行なうということが行なわれている．SNS 上のコミュニティで言語の翻訳の必要が生じた場合，複数のユーザで協力して翻訳をするという状況が考えられる．このことを受け，コミュニティ，特に SNS 上のコミュニティで協調翻訳が行なわれることを想定して研究を行った．

コミュニティ上での会話には，そのコミュニティ特有の単語が存在する場合がある．また，その単語はコミュニティ上で何度も繰り返し使われる可能性が高い．あるコミュニティ特有の単語は機械翻訳で正しく翻訳できない可能性が一般の単語より高いので，コミュニティ上で協調翻訳を行なうと，その単語が修正される可能性が高くなる．したがって，コミュニティ上での協調翻訳から生成される対訳辞書は，そのコミュニティ特有の単語や表現の対訳を多く含んだコミュニティ辞書となる．例えば，京都大学の大学生内で行なわれる協調翻訳からの辞書生成では,「京大翻訳！」のように，京都大学に関連する用語を多

[1] https://twitter.com
[2] https://www.facebook.com
[3] http://wikipedia.org/

く含む可能性が高い．

協調翻訳は一般に専用のツール等を用いて行なわれるために，SNS 上で行なうことを想定する場合には SNS の枠組みのなかで協調翻訳を行なう方法について規定する必要がある．SNS での協調翻訳を SNS の投稿，返信機能を利用して，下記の方法で行なう．

- 投稿者は翻訳したい文章（原文）を投稿する．
- 投稿者は翻訳したい文章に対して機械翻訳を行い，その翻訳文を原文への返信として投稿する．
- 修正作業者は翻訳文が修正可能な場合，原文と翻訳文もしくは翻訳文のみを見て，その翻訳文に対し返信として修正した文章を投稿する．

修正作業は投稿者の投稿を見る全員が可能であるが，基本的には投稿者の友人を指すこととする．上記の手法が実行できる SNS の特徴は以下の通りである．

1. 文章の投稿ができる．
2. 文章の返信ができる．
3. 投稿された文章をタイムラインで見ることができる．

図 4: Twitter 上での協調翻訳

本研究では SNS 協調翻訳が行なわれる一例として Twitter を用いた．実際に図 4 に上記の規則を利用して Twitter 上で協調翻訳を行なった様子を示す．

このようにして，Twitter 上でも協調翻訳作業を行うことが可能である．

3.2　協調翻訳からのコミュニティ辞書構築

本研究での協調翻訳の手法は,「翻訳が修正可能であれば修正する」というものであった．この手法で SNS で協調翻訳を行なった場合，同じ単語に対して，コミュニティのユーザによって様々な修正が行なわれる可能性がある．ユーザによるある単語に対する修正には，コミュニティや専門分野により異なる訳語になるものや，ユーザの勘違いなどによる誤った修正なども含まれる可能性がある．

このように生成された協調翻訳結果から，コミュニティに利用される正しい対訳を抽出するためには以下のような方法が考えられる

1. 最新の単語が最も良いとして抽出する
2. 修正された単語の中で最も多い単語を抽出する
3. 修正された単語についての正確性の投票をユーザによって行ない，多数決により正確な単語を決定し，抽出する．

また，SNS の修正履歴にはアカウント情報が入っており，個人情報が利用可能である．個人ごとの修正の内容の解析などにより，どのようなコミュニティでどのような対訳が使われるかというということや，使う対訳によって個人の所属するコミュニティを判断するといったことができる可能性がある．

第 4 章　協調翻訳からの対訳抽出

前章では，本研究でのシナリオと SNS での協調翻訳の方法について示し, コミュニティ辞書の生成方法についての提案について述べた．本研究では，上記提案手法の一環として，協調翻訳からの対訳抽出手法について設計し，その手法を用いてアプリケーションを実装した．本章では，実装に用いた協調翻訳結果からの対訳抽出手法について説明する．まず，従来の手法との対訳抽出の仕方の違いについて述べる．その後，SNS の協調翻訳での結果から，対訳を抽出する方法について説明する．本研究では，日本語から英語への協調翻訳を想定している．

先行研究における原文と翻訳文からの対訳抽出の研究では，言語情報や統計情報から対訳となる語の候補をあげ，その中から信頼度の高い対訳のペアを辞書に登録することが行なわれている．これらは基本的に対訳コーパス，つまり内容の同じ日本語の文書と英語の文書をもとに対訳を抽出している．つまり，日

本語のある単語が英語でどの単語を表しているかを統計情報などから推定する必要があった．協調翻訳と対訳コーパスからの対訳抽出の違いは，協調翻訳が機械翻訳文を修正していく手法であるということである．原文の単語が機械翻訳文のどの単語であるかという情報は，その機械翻訳が単語をどのように翻訳するかを確認すれば容易に確認が可能である．つまり，原文と翻訳文の単語の対応をもとめるのに統計的情報などを使用する必要がないので，一文だけでも単語の同定が可能である．その後，機械翻訳文を人手により修正していくので．どの単語がどのように修正されたかという情報がわかれば，原文と協調翻訳での翻訳後の文章との単語の対応関係を抽出することができる．

まず，原文と機械翻訳の対応を求めなければならない．機械翻訳の処理内容が参照できる場合はそれを参照するのが一番適当である．本研究では，処理内容を参照できなかったため，単純に再度機械翻訳を使うことで対応を求めた．具体的には次の方法で行なった．まず原文の形態素解析を行なう．形態素解析で分割された単語ごとに機械翻訳をかけていく．機械翻訳の同じ単語に対しての翻訳は常に同じであるから，各単語を機械翻訳にかけた結果が機械翻訳された文章に含まれていれば，その単語と機械翻訳した後の語を記録しておく．このような手順によって，基本的な名詞や形容詞などは抽出可能である．

つぎに，協調翻訳で行なわれた修正の差分を抽出する方法について述べる．本研究では，協調翻訳の修正は単語や連続する語のみとする．つまり，文章の構造全体を変更して新しい文章を作る修正方法については考慮していない．図 5 のような状況の場合には前後の単語の一致を確認することによって，A から A’ への修正を抽出する．

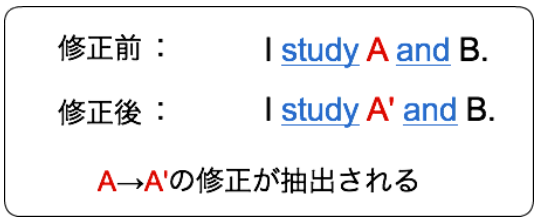

図 5: 修正の検出

これまで原文と機械翻訳文の単語の対応，協調翻訳の修正情報からの差分の抽出について述べた．協調翻訳は何度も修正が起こるため，その差分を履歴とし

て保存し，修正履歴をたどることで修正された対訳を抽出する方法を提案する．

修正の最も簡単な形は，一つの同じ単語が複数回修正される場合である．たとえば“ これは A です ”といった文章の“ A ”の部分だけが修正されていくことを考える．この場合履歴は次のようになる．

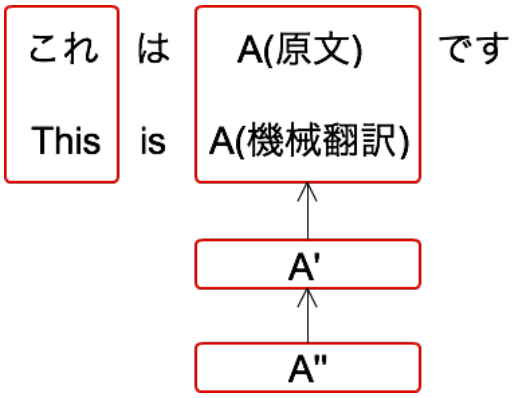

図 6: 一つの単語に対しての修正

この例の場合に履歴として記録されているのは，赤枠で囲まれた部分で，原文と機械翻訳の単語の中で対応のとれた単語と修正された差分の情報である．この場合は一番最新の修正から対応する単語の履歴をたどり原文と機械翻訳が保存されている履歴まで到達すれば，原文と最新の修正の対応をとることが可能となる．この場合を基本形とおくとフローチャートは図 7 のようになる．

最新の修正から，履歴をたどっていき，原文の階層に到達する．原文と機械翻訳の単語のペアとして保存されているものの中で，修正されている単語があればその原文の単語と最新の修正の単語を辞書に登録する．

次に複合名詞に対しての処理について説明する．日本語から英語への機械翻訳において，日本語が複合名詞の場合，その翻訳は誤訳になる可能性が高い．複合名詞とは 2 つもしくはそれ以上の名詞が複合し，一つの名詞となったものである．例えば,“ 協調翻訳 ”という名詞は“ 協調 ”と“ 翻訳 ”という二つの名詞によって構成されており，複合名詞である．

複合名詞が協調翻訳によって修正された場合に前述した単純な履歴を用いた方法を適用すると図 8 のようになる．

複合名詞は形態素解析において，その名詞ごとに単語と見なされるので，それぞれの名詞は独立に翻訳される．この状態で修正が行なわれると図 8 のように，

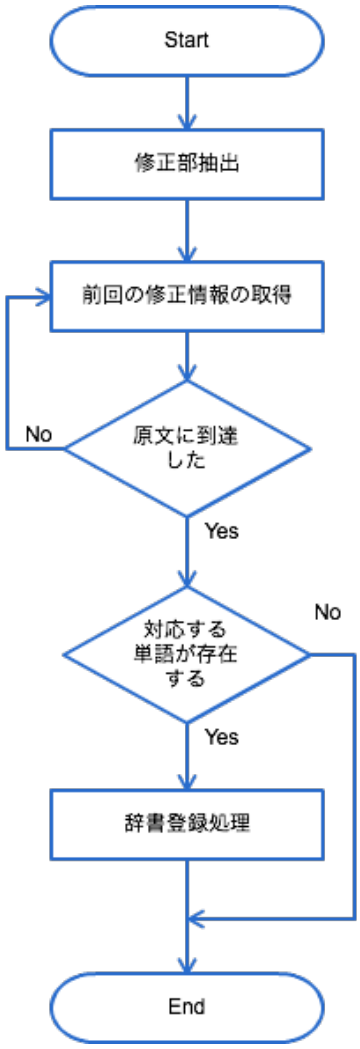

図 7: 履歴からの対訳抽出のフローチャート

実際には“ cooperation translation ”の修正として“ collaborative translation ”という修正が行なわれているのに対して，プログラムは“ cooperation ”に対して“ collaborative ”という修正がなされたと見なし，“ 協調 ”という単語の翻訳として“ collaborative ”という単語が辞書に登録されてしまう．

複合名詞への対処は自然言語処理の研究においても行なわれている [15]．本研究では，単純に形態素解析で連続する名詞が出た場合にはその名詞をあらかじめ結合しておき，履歴に登録しておく．“ 協調翻訳 ”という単語に対しては“ 協調 ”と“ 翻訳 ”二つの単語の機械翻訳が履歴として保存されてしまうが，その二つを“ 協調翻訳 ”という一つの言葉として登録しておくことで，そのうちの一つの単語が修正された場合に，その修正が複合名詞の修正として抽出可能

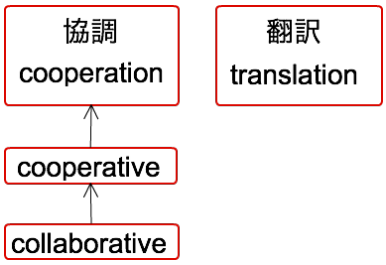

図 8: 複合名詞に対する手法の適用

となる．

最後に，修正が対立した場合の処理について述べる．SNS での協調翻訳作業では図 9 のように修正が対立する状態が生じる可能性がある．

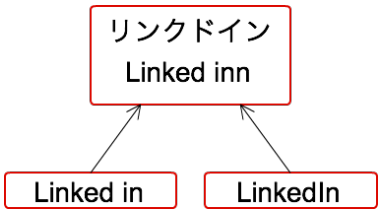

図 9: 協調翻訳での修正の対立

修正の対立が生じる理由としては次のような場合がある．

- 複数の修正者がほぼ同時に修正を投稿した場合
- 複数の修正者が自分の翻訳は正しいが，相手の翻訳文も正しいと感じた場合
- SNS の友達関係などの状態により修正されたことが見えなかった場合

このような修正の対立が生じた場合，2 つとも辞書に登録した状態では，基本的にその対訳辞書を利用した機械翻訳では検索して先に発見した方を対訳として利用してしまうので,“ リンクドイン ”という単語に対しての翻訳は常にどちらか一方になってしまう．つまり，どちらか一つを何らかの手法で評価して選ぶ必要がある．評価方法としては，コミュニティ構成員の投票による多数決などが考えられるが，本研究では協調翻訳は最新のものがもっとも適切な翻訳であるという仮定のもとに，最新の単語を辞書に登録している．

第 5 章　Twitter 協調翻訳支援システム

本章では，Android アプリとして実装した協調翻訳システムと，そのアプリからの投稿を保存し解析するサーバー側のアプリケーションについて説明する．

まずシステム全体の概要について述べ，その後クライアントアプリケーションとサーバーでの処理について説明を行なう．

5.1 システムの概要

図 10 はシステムの全体の概要を表したものである．

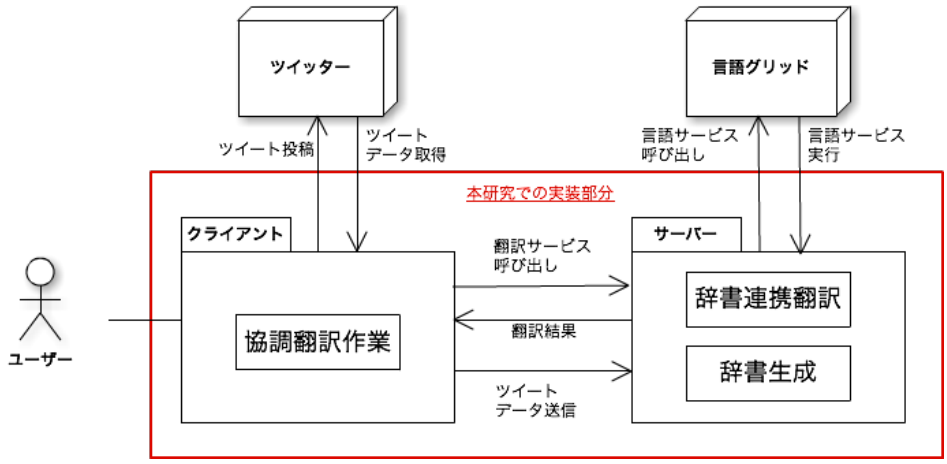

図 10: システムの概要図

協調翻訳作業はクライアントアプリケーションとして実装し，ユーザーはクライアントアプリケーション上で協調翻訳を行なう．協調翻訳作業は全て Twitter 上に投稿されるようになっている．クライアントアプリケーションは投稿された Twitter のステータス情報を逐次サーバーに送り，サーバーのデータベースに保存されるようになっている．これは Twitter の API には利用回数制限があるため，サーバーでのツイートの解析時に障害となるためである．今回の実装ではクライアントアプリケーションはサーバーのデータベースからツイートの情報を取得しているため，クライアントアプリケーション上のタイムラインの表示は実際の Twitter のものとは異なり，本研究のクライアントアプリケーションを利用した投稿のみが表示されるようになっている．

サーバーではデータベースに保存されたツイートのステータス情報を使った解析処理を行ない，修正情報をデータベースに保存する．そして，修正情報をたどることで対訳を抽出し，辞書用のデータベースに登録しておく．クライアントアプリケーションから辞書を利用した機械翻訳の呼び出しがあった場合，辞書用のデータベースを利用した言語グリッドの辞書連携翻訳を呼び出し，結果

をクライアントアプリケーションに送信する．

5.2　クライアントアプリのユーザーインターフェイスと実行例

5.2.1　ユースケース

クライアントアプリケーションのユースケース図を図 11 に示す．クライアントアプリケーションは Android アプリとして実装し，下記の機能を持つ．

- ツイッターへのログイン
- ツイッターへの投稿
- タイムラインの表示
- タイムライン上の投稿への修正（リプライ）
- 言語選択 (日本語もしくは英語)

ユーザーには投稿者と修正作業者があり，投稿者は文章の投稿を行ない，その文章が自動的に機械翻訳され同時に投稿される．修正作業者はリプライ機能を使って投稿者が投稿した文章の機械翻訳文を修正する．投稿者自身によって機械翻訳文を修正することも可能である．

5.2.2　ユーザーインターフェイス

図 12 はクライアントアプリのユーザーインターフェースである．以下図 12 の構成を説明する．

1. 言語選択ボタン
   ここでユーザーは自分の使用する言語を選択する．例えば，Japanese を選択すると日本語を投稿することで機械翻訳により英語に翻訳される．
2. 投稿ボックス
   このテキストボックスに文章を入力し Tweet ボタンを押すことで文章を投稿することができる．左下の数字はツイッターの投稿の文字制限である．このアプリは Twitter 上で動作するように作られており，Twitter の文字制限をオーバーした場合には警告を表示し，投稿しない．また，投稿された文章は自動的に機械翻訳され，自分へのリプライとして投稿される．この際も，Twitter の文字制限をオーバーすると警告を表示し原文，翻訳文ともに投稿されないようになっている．
3. タイムライン
   タイムラインでは自分や他のユーザーが投稿した文章が表示される．誰かが文章の投稿を行なうと，タイムラインに自動的に順次表示されるように

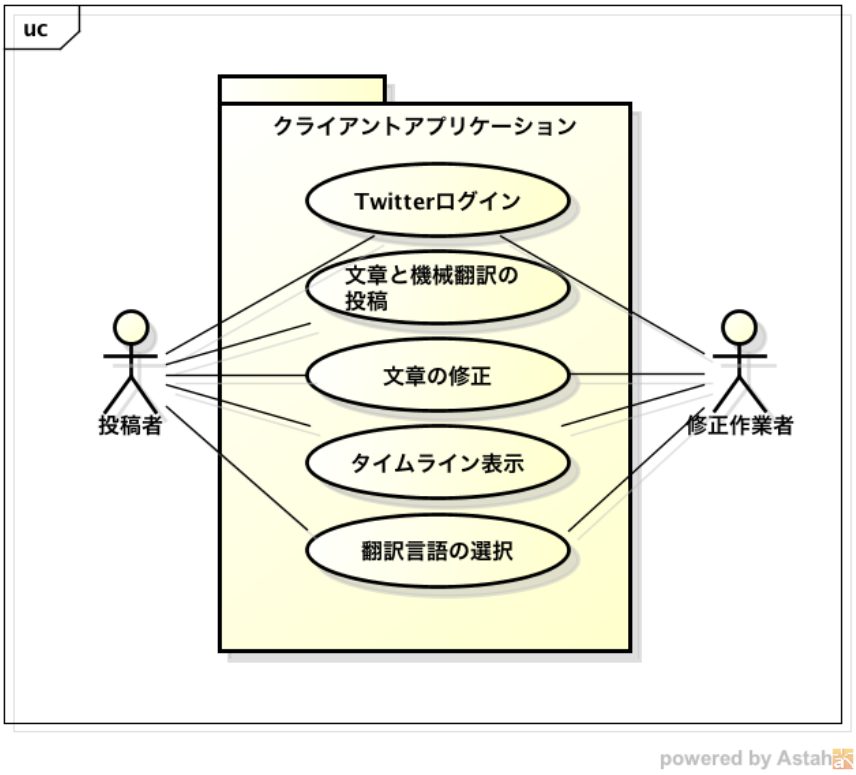

図 11: クライアントアプリのユースケース図

なっている．UPDATE ボタンは手動でタイムラインの全ての情報を更新するボタンである．タイムライン上の投稿をクリックすることで，その投稿の修正画面が表示される．

ここからタイムライン上のアイテムをクリックしたときに表示される修正画面について説明する．修正画面が表示されたときのスクリーンショットは図 13 に示している．

1. 修正ボックス

   修正ボックスにはタイムラインでクリックした投稿の文章が自動的に入力されている．ユーザーがタイムライン上の文章を修正したい場合はこのボックス内にある文章を部分的もしくは全体を削除して reply ボタンを押すことで修正の投稿が完了する．修正はツイッターでのリプライとして投稿される．

2. 原文表示ボックス

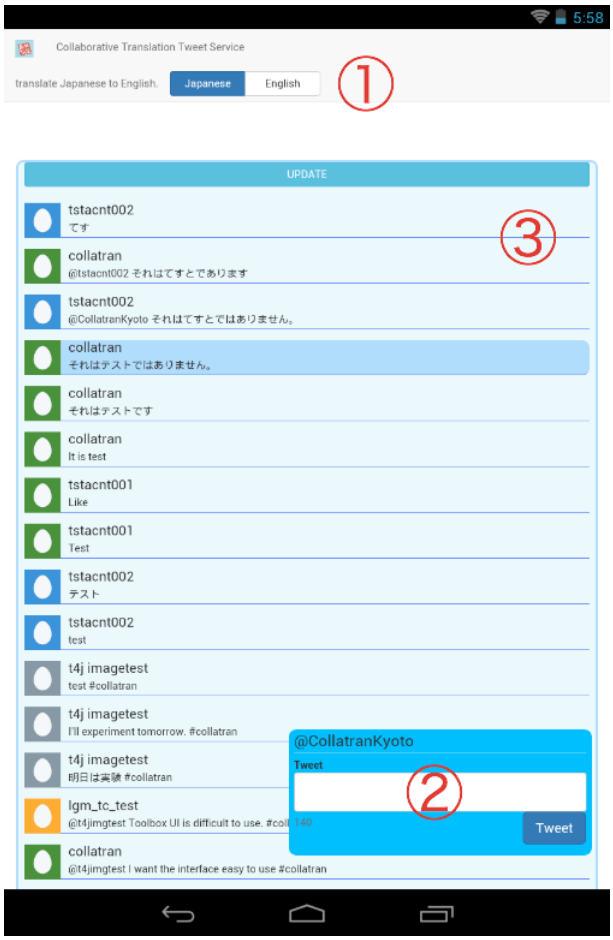

図 12: クライアントアプリのスクリーンショット

原文表示ボックスには修正の投稿や機械翻訳の原文が表示される．原文をクリックした場合は何も表示されない．

5.2.3 プログラムの構成

クライアントアプリケーションの主な機能についてのクラス図を図 14 に示した．本実験では実装に Apache-Cordova[1] を利用しているため，ユーザーインターフェイスは html や Javascript で記述されている．Twitter の処理は twitter4j[2] を利用しており，翻訳関連の処理とともに Java で書かれている．

5.3 サーバーの実装

サーバーアプリケーションのデータの流れを図 15 に示す．

[1] http://cordova.apache.org
[2] http://twitter4j.org/ja/

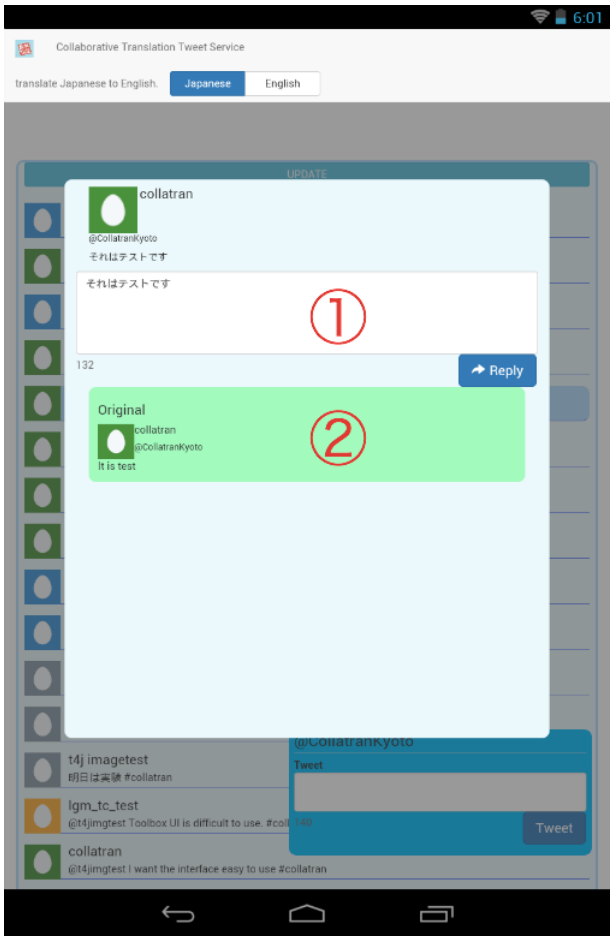

図 13: クライアントアプリの修正画面のスクリーンショット

サーバーでは主に下記の機能が実装されている．

- ツイートデータの保存
- 原文と機械翻訳文の単語の対応を求める処理
- ツイートデータからの修正差分の抽出
- 差分を保存した履歴データからの対訳抽出

サーバーでの処理は以下の流れで行なわれる．

1. ツイート保存サービス

   ツイート保存サービスはクライアントから送信されるツイートのステータスデータをツイート保存用データベースに保存する．保存するデータとしては，Twitter のステータスに元々含まれるツイートの ID，テキスト，リプライしたツイートの ID，ユーザネームといったものに加え，クライアントアプリ上で設定した言語選択から，そのツイートの言語の情報を追加保

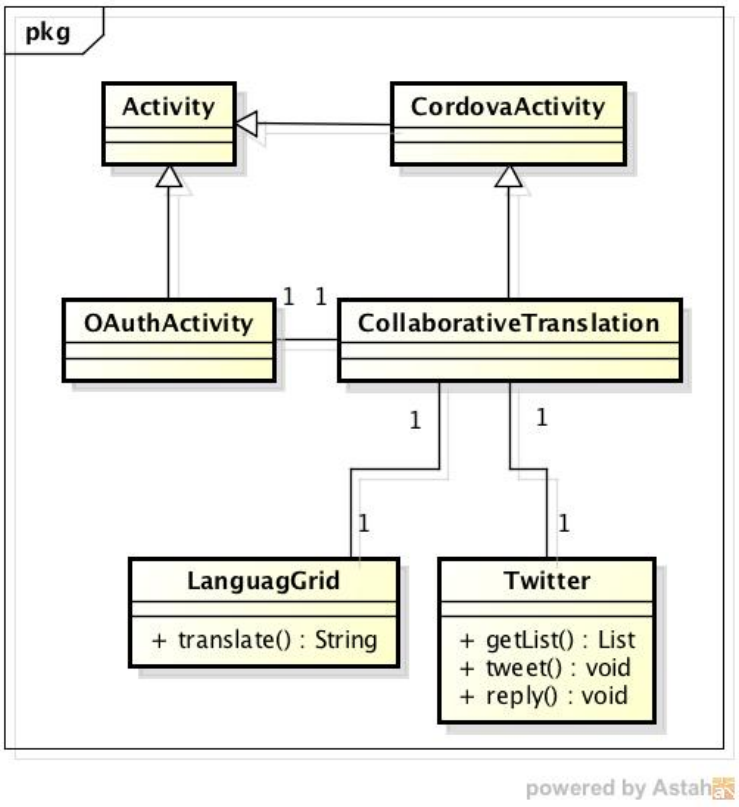

図 14: クライアントアプリのクラス図

存している．

2. 修正抽出処理
   修正抽出処理では，前章で説明した原文と機械翻訳文との単語の対応と，ツイートの文章とそのツイートが返信されたツイートの文章の差分の検出処理を行なう．差分の検出処理後，単語単位で修正履歴保存用データベースに保存しておく．
3. 対訳抽出サービス
   対訳抽出処理ではデータベースに保存されている修正情報のリプライ情報をたどることで対訳を抽出し，対訳辞書データベースに保存する．
4. 辞書連携機械翻訳サービス
   辞書連携機械翻訳サービスは言語グリッドのサービスを利用している，対訳辞書データベースに含まれる対訳を優先的に利用する機械翻訳を利用できるサービスである．

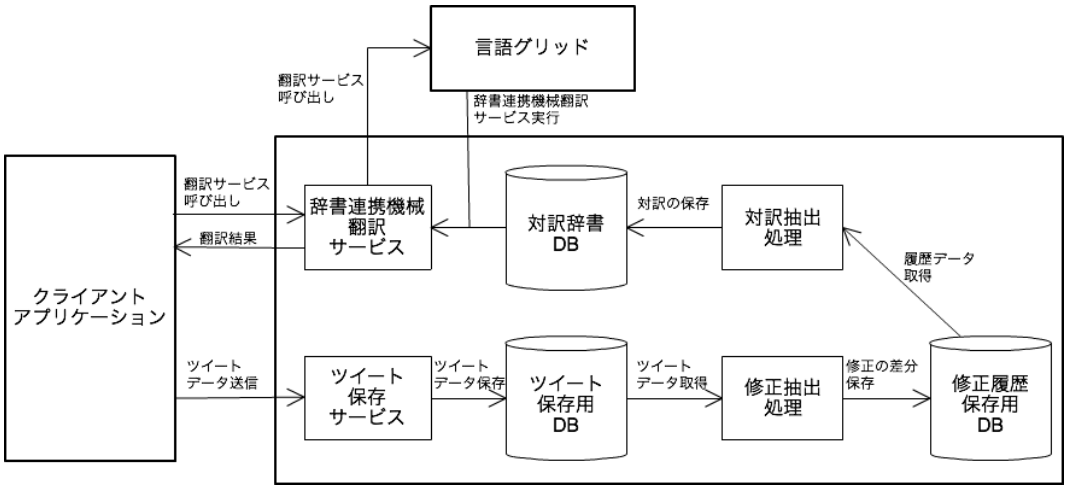

図 15: サーバーでの主なデータの流れ

第6章　実験と考察

6.1　実験手法と目的

実装した協調翻訳アプリケーションを使って，実際に協調翻訳を行なう実験を行なった．実験の目的は，特定の会話のテーマに沿った単語が実際に協調翻訳から抽出されるかどうかの検証である．そして，現時点での手法の問題点についての知見を得ることを目的とする．参加者は日本語話者が 3 人と英語話者 1 人である．日本語話者は英語の読み書きが可能であり，英語話者は日本語の知識を持たない．実験の手順は以下の手順で行なった．

1. まず特定のテーマ「最近興味のある技術」について協調翻訳アプリケーション内で自由に話を行なってもらう（30 分間）
2. その後，自由に話をした内容が自動的に機械翻訳された文章に間違いがある場合に修正を行なってもらう（20 分間）
3. さらにその後，修正が可能な他の作業者の修正文をを修正してもらう（5 分間）

日本語話者三人は英語と日本語の両方を理解できるので，修正の際は日本語の原文と英語の機械翻訳文両方を見ながら修正を行なってもらった．英語話者の大学院生は日本語がほとんど理解できないため，機械翻訳の英語のみを見てもらい，できる範囲で単語や表現の修正を行なってもらった．ただし，実装したアルゴリズムでは複雑な修正には対応できないため修正には次の制約を設けた．

- 単語の移動は行なわない

- 文章の構文自体を変更しない

6.2 実験結果と考察

以上で説明した手順，手法によって実際に協調翻訳の修正作業を行なってもらった結果を以下に示す．まず，対訳として抽出された単語として表 2 のものがあった．

表 2: 実験での対訳抽出部分

原文での単語	機械翻訳	修正
リンクドイン	Linked inn	Linked in
		LinkedIn
機械学習	mechanical learning	machine learning

表 2 の対訳は「最近興味のある技術」というテーマからの会話から抽出された対訳であり，そのテーマに見合った単語である．このことから，あるテーマの会話での協調翻訳結果から，そのテーマに関連した単語の対訳を抽出可能であることがわかった．

提案手法の課題を探るため，実験後参加者にインタビューを行なった．参加者からの意見は以下の通りである．

- 単語のみで修正できる文章はほとんどない
- 単語だけの修正だと不十分で英語話者は全く会話が理解できない
- 英語話者は機械翻訳のみでは理解できず修正できない
- 文章がわからない原因は構造がずれていることがほとんど
- どの文章が修正されたかわからない
- リプライがどんどん上に出てくるので会話の理解が進まない

今回の実験では，はじめに自由に会話を行なってもらった．その際，日本語話者 3 人の会話は日本語で行なわれた．その日本語は，アプリケーションによって英語に機械翻訳され，英語話者はその機械翻訳から会話の意味を読み取ることになる．はじめがフリートークであった点もあり，普通の会話のような口語表現も多かったため，正確な機械翻訳ができず英語話者はその意味をほとんど理解することができなかった．実際，フリートークでの全発話数 34 回に対して，

英語話者の発言は1回のみであった．また，協調翻訳の修正は本来英語話者でも内容が理解できるように人手によって機械翻訳をサポートする物であるが，今回はフリートークと修正作業を分けてしまったために協調翻訳の修正が生かされなかった．

本研究では修正の差分を求めるアルゴリズムが単語のみの修正のみを想定しているため，実験でも構文は修正せずに単語のみの修正を行なってもらうようにしたが,「文章がわからない原因は構造がずれていることがほとんど」という意見にもあるように，機械翻訳が分からない原因としては，本研究で解決しようとしている専門用語の誤訳の他にも，翻訳された文書の文法構造自体の不正確性が上げられる．これに対して，単語のみの修正という制約をつけてしまうと，協調翻訳によって人手によって修正している意味が無くなってしまう．つまり，単語の修正だけでなく文章自体の構成を変更したとしても，その修正の差分を抽出し，単語の対応を求めることのできるような手法が必要である．

第7章　おわりに

本研究では，機械翻訳の精度を補うために，複数の作業者により翻訳を繰り返し修正していく協調翻訳という手法について紹介した．その協調翻訳という手法に対して，その結果が再利用されないために，同じ文章を再度翻訳しなければならないという問題に対して取り組んだ．この問題に対してまず，コミュニティでの協調翻訳という状況について想定し，その状況に対してSNS上での協調翻訳手法について提案した．協調翻訳結果からの対訳抽出手法として以下のような手順で行なう手法を提案した．

1. 原文と機械翻訳文の単語の対応関係の同定
   まず，原文に形態素解析を行い，再度機械翻訳を行なうことで原文と機械翻訳文の単語の対応を求める．
2. 修正の差分抽出
   修正前の文章と修正後の文章の差分を前後の単語の一致により求め，修正履歴として保存する．
3. 修正履歴からの対訳抽出
   保存された修正履歴をたどることで最終的な協調翻訳結果と原文との単語の対応を求め，辞書に登録する．

本研究では以上の手法に基づいた協調翻訳システムを実装した．本研究の貢献は以下の二点である．

1. SNS で協調翻訳を行なう手法の設計
   本研究ではSNS で協調翻訳を行なうための手法として原文と機械翻訳文を同時に投稿し，機械翻訳文に修正が必要な場合に返信機能を使って修正を行なうという手法を提案した．
2. 抽出した対訳を機械翻訳に利用するシステムの設計と実装
   協調翻訳作業を SNS 上で行なうことのできるクライアントアプリケーションを実装した．上記の手法により対訳を抽出しその対訳の辞書への登録および対訳辞書を用いた機械翻訳が利用可能なサーバーアプリケーションを実装した．

実験により明らかになった今後の課題として，現在の手法では機械翻訳の文法上の間違いに対応できないという問題がある．本研究の手法は新語や専門用語などの単語のみの修正において有効であるが，協調翻訳の本来の目的である機械翻訳文の滑らかさを上げるために構文自体を変えてしまうような修正には対応できていない．新しい修正の差分の抽出手法の設計により，構文が変わっても単語の差分が抽出できるようにすることが必要である．

謝辞

研究の機会を与えてくださり，熱心なご指導，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，日頃より時間を惜しまず様々なご助言，ご協力をいただきました松原繁夫准教授，大谷雅之特定研究員，中口孝雄特定研究員，林冬惠特定助教に感謝申しあげます．最後に，後藤真介先輩をはじめ，普段から数々の助言をいただき，実験にもご協力いただきました研究室の皆様に心より感謝いたします．

参考文献

[1] Douglas, S. and Craig, C.: Collaborative and iterative translation: an alternative approach to back translation, *Journal of International Marketing*, Vol. 15, No. 1, pp. 30–43 (2007).

[2] Shimohata, S., Kitamura, M., Sukehiro, T. and Murata, T.: Collaborative

Translation Environment on the Web, *Proceedings of the MT Summit VIII*, pp. 331–334 (2001).

[3] Ishida, T.: *The Language Grid: Service-Oriented Collective Intelligence for Language Resource Interoperability.*, Springer (2011).

[4] 重信智宏, 藤井薫和, 宮部真衣, 藤原義功, 吉野孝: コミュニティ指向の異文化コラボレーションツールの開発, 第 5 回情報科学技術フオーラム (FIT2006) 情報科学技術レターズ, Vol. 5, pp. 317–320 (2006).

[5] 三原宏一朗, 境智史, Heeryon, C., 石田亨: 多言語コミュニケーションのためのチャットツールの開発, 電子情報通信学会技術研究報告, Vol. 108, No. 441, pp. 79–84 (2009).

[6] 宮部真衣, 吉野孝, 重野亜久里: 外国人患者のための用例対訳を用いた多言語医療受付支援システムの構築, 電子情報通信学会論文誌 D, Vol. 92, No. 6, pp. 708—718 (2009).

[7] 福島拓, 吉野孝, 重野亜久里: 用例対訳を用いた多言語問診票作成システムの開発と評価, 情報処理学会研究報告, Vol. 2011-GN-78, No. 14, pp. 1–7 (2011).

[8] 熊野明, 平川秀樹: 対訳文書からの機械翻訳専門用語辞書作成, 情報処理学会論文誌, Vol. 35, No. 11, pp. 2283–2290 (1994).

[9] 高尾哲康, 富士秀, 松井くにお: 対訳テキストコーパスからの対訳語情報の自動抽出, 情報処理学会研究報告, Vol. 96, No. 87, pp. 51–58 (1996).

[10] 北村美穂子, 松本裕治: 対訳コーパスを利用した対訳表現の自動抽出, 情報処理学会論文誌, Vol. 38, No. 4, pp. 727–736 (1997).

[11] 梶博行, 相薗敏子: 共起語集合の類似度に基づく対訳コーパスからの対訳語抽出, 情報処理学会論文誌, Vol. 42, No. 9, pp. 2248–2258 (2001).

[12] 森下洋平, 宇津呂武仁, 山本幹雄: 対訳特許文書からの専門用語対訳辞書半自動獲得におけるフレーズテーブルと既存対訳辞書の併用, 情報処理学会自然言語処理研究会, NL-187, pp. 91–98 (2008).

[13] Zaidan, O. and Callison-Burch: Crowdsourcing translation: Profes- sional quality from non-professionals, *roceedings of the 49th Annual Meet- ing of the Association for Computational Linguistics: Human Language Technologies*, pp. 1220–1229 (2011).

[14] Morita, D. and Ishida, T.: Collaborative translation by monolinguals with

machine translators, *Proceedings of the 14th international conference on Intelligent user interfaces, ACM*, pp. 361–366 (2009).

[15] 中川裕志, 森辰則, 湯本紘彰: 出現頻度と連接頻度に基づく専門用語抽出, 自然言語処理, Vol. 10, No. 1, pp. 27–45 (2003).

付録：主なソースコード

A.1 クライアントアプリケーション

A.1.1 CollaborativeTranslation.java

```
package org.langrid.collatran;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import org.apache.cordova.Config;
import org.apache.cordova.CordovaActivity;
import org.json.JSONObject;
import android.webkit.JavascriptInterface;
import android.webkit.WebView;
import android.content.SharedPreferences;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.util.Log;
import android.app.Activity;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.content.Context;
import android.content.ContentValues;

public class CollaborativeTranslation
extends CordovaActivity {
 public static final String PREF_NAME =
 "collatran_preference";
 public static final String TOKEN = "token";
 public static final String TOKEN_SECRET = "token_secret";
 private String token;
 private String tokenSecret;

@Override
public void onCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
super.init();
appView.addJavascriptInterface(this,"MyCls");
//login function
SharedPreferences preferences =
getSharedPreferences(PREF_NAME,MODE_PRIVATE);
token = preferences.getString(TOKEN, null);
 tokenSecret = preferences.getString(TOKEN_SECRET, null);
 if (token == null || tokenSecret == null ) {
```

```
 Intent intent = new Intent(this, OAuthActivity.class);
 startActivity(intent);
 finish();
 }
 preferences.edit().clear().commit();
 super.loadUrl(Config.getStartUrl());
}
@JavascriptInterface
public String getToken(){
return token;
}
@JavascriptInterface
public String getTokenSecret(){
return tokenSecret;
}
}
```

A.1.2 OAuthActivity.java

```
package org.langrid.collatran;
import java.net.URI;
import java.net.URL;
import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.ViewGroup;
import android.widget.Toast;
import twitter4j.*;
import twitter4j.auth.AccessToken;
import twitter4j.auth.OAuthAuthorization;
import twitter4j.auth.RequestToken;
import twitter4j.conf.Configuration;
import twitter4j.conf.ConfigurationContext;

public class OAuthActivity extends Activity {
 private OAuthAuthorization mOAuth = null;
 private RequestToken mReq = null;
 public static final String CALLBACK_URL = "";
 private final int FP = ViewGroup.LayoutParams.MATCH_PARENT;
 private final int WC = ViewGroup.LayoutParams.WRAP_CONTENT;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_oauth);
new OAuthTask().execute();
Toast.makeText(this, "log in to Twitter",
Toast.LENGTH_SHORT).show();
}
@Override
protected void onNewIntent(Intent intent) {
super.onNewIntent(intent);
new TokenTask().execute(intent);
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item)
{
int id = item.getItemId();
return super.onOptionsItemSelected(item);
}
public class OAuthTask
extends AsyncTask<Void,Void,Void> {
@Override
protected Void doInBackground(Void... arg0) {
Configuration conf = ConfigurationContext.getInstance();
mOAuth = new OAuthAuthorization(conf);
mOAuth.setOAuthConsumer(getString(R.string.consumer_key),
getString(R.string.consumer_secret));
try {
mReq = mOAuth.getOAuthRequestToken(CALLBACK_URL);
} catch (TwitterException e) {
e.printStackTrace();
return null;
}
String uri;
uri = mReq.getAuthorizationURL();
startActivity(new Intent(Intent.ACTION_VIEW,
Uri.parse(uri)));
return null;
}
}
```

```
 public class TokenTask
 extends AsyncTask<Intent,Void,AccessToken>{
 @Override
 protected AccessToken doInBackground(Intent... params) {
 Intent intent = params[0];
 Uri uri = intent.getData();
 AccessToken token =null;
 if (uri != null && uri.toString()
 .startsWith(CALLBACK_URL)) {
 String verifier = uri.getQueryParameter("oauth_verifier");
 try {
 token = mOAuth.getOAuthAccessToken(mReq, verifier);
 } catch (Exception e) {
 e.printStackTrace();
 }
 }
 return token;
 }
 @Override
 protected void onPostExecute(AccessToken result) {
 if (result != null) {
 SharedPreferences preferences =
 getSharedPreferences(CollaborativeTranslation.PREF_NAME,
 MODE_PRIVATE);
 SharedPreferences.Editor editor = preferences.edit();
 editor.putString(CollaborativeTranslation.TOKEN,
 result.getToken());
 editor.putString(CollaborativeTranslation.TOKEN_SECRET,
 result.getTokenSecret());
 editor.commit();
 Intent sIntent = new Intent(getApplicationContext(),
 CollaborativeTranslation.class);
 startActivity(sIntent);
 }
 finish();
 }
 }
}
```

A.1.3 LanguageGrid.java

```
package org.langrid.collatran.plugin;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import org.langrid.client.Client;
```

```
import org.langrid.client.soap.SoapClientFactory;
import org.langrid.lamshup.android.common.LGMPlugin;
import org.langrid.lamshup.android.common.annotation.PluginMethod;
import org.langrid.service.api.common.InvalidParameterException;
import org.langrid.service.api.common.ProcessFailedException;
import org.langrid.service.api.language.Translation;
import org.langrid.service.api.language.TranslationService;
import org.langrid.service.api.language.LanguageIdentificationService;
import java.util.*;

public class LanguageGrid extends LGMPlugin {
private static final String jserver_url =
"http://langrid.org/service_manager/invoker
/kyoto1.langrid:KyotoUJServer";
private final String id = "";
private final String pass = "";

@PluginMethod
public String translate(String sourceLang,
String targetLang, String source)
throws URISyntaxException, InvalidParameterException,
ProcessFailedException {
Client client = new SoapClientFactory()
.createClient(
new URI(jserver_url),
id, pass);
return client.proxy(TranslationService.class)
.translate(sourceLang,targetLang, source);
}
@PluginMethod
public String identifyLanguage(String text)
throws URISyntaxException,
InvalidParameterException, ProcessFailedException {
Client client = new SoapClientFactory()
.createClient(
new URI(identify_url),
id, pass);
return client.proxy(LanguageIdentificationService.class)
.identify(text,"utf-8");
}
@PluginMethod
public String autoTranslate(String targetLang, String text)
throws InvalidParameterException,
ProcessFailedException,
URISyntaxException {
```

```
String sourceLang = identifyLanguage(text);
return translate(sourceLang, targetLang, text);
}
}
```

A.1.4 ExpandedStatus.java

```
package org.langrid.collatran.plugin;
import org.langrid.lamshup.android.common.LGMPlugin;
import twitter4j.*;

public class ExpandedStatus extends LGMPlugin{
Status status;
String id;
String inReplyToStatusId;

public String getInReplyToStatusId() {
return inReplyToStatusId;
}
public void setInReplyToStatusId(String inReplyToStatusId) {
this.inReplyToStatusId = inReplyToStatusId;
}
public Status getStatus() {
return status;
}
public void setStatus(Status status) {
this.status = status;
}
public String getId() {
return id;
}
public void setId(String id) {
this.id = id;
}
public ExpandedStatus(Status status) {
super();
this.status = status;
this.id = ((Long)status.getId()).toString();
this.inReplyToStatusId =
((Long)status.getInReplyToStatusId()).toString();
}
public ExpandedStatus(Status status, String id) {
super();
this.status = status;
this.id = id;
}
```

```
}
```

A.1.5 Twitter.java

```
package org.langrid.collatran.plugin;
import org.langrid.lamshup.android.common.LGMPlugin;
import twitter4j.*;
import twitter4j.auth.AccessToken;
import twitter4j.auth.OAuthAuthorization;
import twitter4j.auth.RequestToken;
import twitter4j.conf.Configuration;
import twitter4j.conf.ConfigurationBuilder;
import twitter4j.conf.ConfigurationContext;
import jp.go.nict.langrid.client.jsonrpc.*;
import java.util.*;
import java.net.MalformedURLException;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import org.langrid.client.Client;
import org.langrid.client.soap.SoapClientFactory;
import org.langrid.collatran.CollaborativeTranslation;
import org.langrid.collatran.datastore.api.MatchingStoreService;
import org.langrid.collatran.datastore.api.Tweet;
import org.langrid.collatran.datastore.api.TweetStoreService;
import org.langrid.lamshup.android.common.LGMPlugin;
import org.langrid.lamshup.android.common.annotation.PluginMethod;
import org.langrid.service.api.common.InvalidParameterException;
import org.langrid.service.api.common.ProcessFailedException;
import org.langrid.service.api.language.TranslationService;
import android.R.integer;
import android.content.Intent;
import android.net.Uri;
import android.util.Log;
import android.content.SharedPreferences;
import android.content.ContextWrapper.*;

public class Twitter extends LGMPlugin {
private twitter4j.Twitter twitter;
private String queryString;
private static final String CONSUMER_KEY = "";
private static final String CONSUMER_SECRET = "";
static final String address = ""; // server's IP address
static final String url = "http://" + address
+ ":8080/CollatranDataStore/jsServices/TweetStore";
static final String url_matching = "http://" + address
```

```
+ ":8080/CollatranDataStore/jsServices/MatchingStore";
private static List<ExpandedStatus> streamList =
new ArrayList<ExpandedStatus>();
private static List<ExpandedStatus> streamList2 =
new ArrayList<ExpandedStatus>();
private static boolean streamFlag = true;
private int listlen=0;

@PluginMethod
public void init(String token, String tokensecret)
throws TwitterException {
id = 0;
twitter = TwitterFactory.getSingleton();
try {
twitter.setOAuthConsumer(CONSUMER_KEY,
CONSUMER_SECRET);
} catch (IllegalStateException e) {
Log.d("setComsumerkey", e.toString());
} finally {
twitter.setOAuthAccessToken(
new AccessToken(token, tokensecret));
streamList = new ArrayList<ExpandedStatus>();
}
}
static class MyStatusListener
implements StatusListener {
@Override
public void onException(Exception arg0) {
}
@Override
public void onDeletionNotice(StatusDeletionNotice arg0) {
}
@Override
public void onScrubGeo(long arg0, long arg1) {
}
@Override
public void onStallWarning(StallWarning arg0) {
}
@Override
public void onStatus(Status status) {
Log.d("STREAM", "Status" + status.getText());
if (streamFlag) {
streamList.add(new ExpandedStatus(status));
} else {
streamList2.add(new ExpandedStatus(status));
```

```
}
}
@Override
public void onTrackLimitationNotice(int arg0) {}
}
@SuppressWarnings("finally")
@PluginMethod
public Status tweet(String text)
throws TwitterException {
Status ret = twitter.updateStatus(text);
try {
new JsonRpcClientFactory()
.create(TweetStoreService.class,
new URL(url)).add(toTweet(ret));
} catch (MalformedURLException e) {
e.printStackTrace();
} finally {
return ret;
}
}
@PluginMethod
public void collaborativeTweet(String text,
String text2, String lang1,String lang2)
throws TwitterException, MalformedURLException {
Status s = twitter.updateStatus(text);
new JsonRpcClientFactory()
.create(TweetStoreService.class,
new URL(url)).add(toTweet(s));
Status st = twitter.updateStatus(
new StatusUpdate(text2)
.inReplyToStatusId(s.getId()));
new JsonRpcClientFactory()
.create(TweetStoreService.class, new URL(url)).add(toTweet(st));
}
@PluginMethod
public List<Status> getTimeLine()
throws TwitterException {
List<Status> ret = twitter.getHomeTimeline();
timelineMaxId = ret.get(ret.size() - 1).getId();
addList(ret);
return ret;
}
@PluginMethod
public List<Status> getTimeLineByMaxId()
throws TwitterException {
```

```
Paging page = new Paging();
page.setMaxId(timelineMaxId);
List<Status> ret = twitter.getHomeTimeline(page);
ret.remove(0);
timelineMaxId = ret.get(ret.size() - 1).getId();
addList(ret);
return ret;
}
@PluginMethod
public void retweet(int index) throws TwitterException {
twitter.retweetStatus(tweetList.get(index));
}
@PluginMethod
public void reply(int index, String replyText)
throws TwitterException,
MalformedURLException {
long replyId = tweetList.get(index);
Status s = twitter.updateStatus(new StatusUpdate(replyText)
.inReplyToStatusId(replyId));
new JsonRpcClientFactory()
.create(TweetStoreService.class, new URL(url)).add(toTweet(s));
}
@PluginMethod
public void reply2(String id, String replyText, String text)
throws TwitterException, MalformedURLException,
InvalidParameterException, ProcessFailedException,
URISyntaxException {
long replyId = Long.parseLong(id);
Status status = twitter.updateStatus(
new StatusUpdate(replyText)
.inReplyToStatusId(replyId));
new JsonRpcClientFactory()
.create(TweetStoreService.class, new URL(url)).add(
toTweet(status));
}
@PluginMethod
public List<Status> getConversation(int index)
throws TwitterException {
return getConversationList(tweetList.get(index));
}
public List<Status> getConversationList(Long id)
throws TwitterException {
List<Status> ret = new ArrayList<Status>();
Status s = twitter.showStatus(id);
while (s.getInReplyToStatusId() > 0) {
```

```
s = twitter.showStatus(s.getInReplyToStatusId());
ret.add(s);
}
return ret;
}
@PluginMethod
public List<Status> search(String query)
throws TwitterException {
queryString = query;
Query q = new Query();
q.count(10);
q.setQuery(query);
List<Status> ret = twitter.search(q).getTweets();
searchMaxId = ret.get(ret.size() - 1).getId();
addList(ret);
return ret;
}
@PluginMethod
public List<Status> getSearchByMaxId()
throws TwitterException {
Query q = new Query();
q.setMaxId(searchMaxId);
q.setQuery(queryString);
q.setCount(30);
List<Status> ret = twitter.search(q).getTweets();
ret.remove(0);
searchMaxId = ret.get(ret.size() - 1).getId();
addList(ret);
return ret;
}
private Tweet toTweet(Status status) {
String tweetId = "";
String text = status.getText();
String statusId = ((Long) status.getId()).toString();
String userId = ((Long) status.getUser().getId()).toString();
String userName = status.getUser().getName();
String screenName = status.getUser().getScreenName();
String inReplyToStatusId =
((Long) status.getInReplyToStatusId()).toString();
String imageURL = status.getUser().getProfileImageURL();
return new Tweet(tweetId, statusId, text,
userId, userName, screenName,
inReplyToStatusId, imageURL);
}
}
```

A.1.6 tweet.html

```
<!DOCTYPE html>
<!--
/*
 * Copyright (C) The Language Grid Project.
 * All rights reserved.
 */
-->
<html>
<head>
<meta charset="utf-8" />
<meta name="format-detection" content="telephone=no" />
<!-- WARNING: for iOS 7,
remove the width=device-width and
height=device-height attributes.
See https://issues.apache.org/jira/browse/CB-4323 -->
<meta name="viewport"
content="user-scalable=no, initial-scale=1,
maximum-scale=1, minimum-scale=1,
width=device-width, height=device-height,
target-densitydpi=device-dpi" />
<link rel="stylesheet" href="css/common.css" />
<link rel="stylesheet"  href="css/index.css" />
<link rel="stylesheet"  href="css/bootstrap.min.css" />
<link rel="stylesheet"  href="css/tweet.css" />
<title>Collaborative Translation</title>
<script type="text/javascript" src="js/jquery-1.11.0.min.js">
<script type="text/javascript" src="cordova.js">
<script type="text/javascript" src="js/app.js">
<script type="text/javascript" src="js/bootstrap.min.js">
<script type="text/javascript" src="js/tweet.js">
</head>
<body>
<nav class="navbar navbar-default navbar-fixed-top"
role="navigation">
<div class="container-fluid">
<div class="navbar-header">
 <button type="button"
 class="navbar-toggle collapsed" data-toggle="collapse"
data-target="#bs-example-navbar-collapse-1">
 <span class="sr-only">Toggle navigation</span>
 <span class="icon-bar"></span>
 <span class="icon-bar"></span>
 <span class="icon-bar"></span>
```

```
 </button>
<a class="navbar-brand" href="#">
<img alt="Brand" src="icon.png"></a>
<p class="navbar-text">Collaborative
Translation Tweet Service</p>
</div>
</div>
 <div class="collapse navbar-collapse"
 id="bs-example-navbar-collapse-1">
 <ul class="nav navbar-nav navbar-right">
 </ul>
 <p class="navbar-text" id="navtext">
 which is your langage?</p>
 <div class="btn-group" role="group" aria-label="...">
 <button type="button" id="ja-button"
 class="btn btn-default navbar-btn">
 Japanese</button>
 <button type="button" id="en-button"
 class="btn btn-default navbar-btn">
 English</button>
</div></div></nav>
<div id="col-center">
<div id="timelineUpdate" class="btn btn-info">UPDATE </div>
<ul class="media-list" id="timeline_list">
<li class="media" id="sample"><a class="media-left pull-left"
href="#"> <img src="star-half.png" alt="...">
</a>
<div class="media-body">
<h4 class="media-heading">Media heading1</h4>
...
</div></li>
<li class="media" id="sample">
<a class="media-left pull-left"
href="#"> <img src="star-half.png" alt="...">
</a>
<div class="media-body">
<h4 class="media-heading">Media heading2</h4>
...
</div></li>
<li class="media" id="sample">
<a class="media-left pull-left"
href="#"> <img src="star-half.png" alt="...">
</a>
<div class="media-body">
<h4 class="media-heading">Media heading3</h4>
```

```
...
</div></li>
<li><div id="timeline-button" >TIMELINE</div></li>
</ul>
</div>
<div id="modalbox">
<div id="col-right">
<div id="tweet-area">
<div id="user">
<img alt="img" src="icon.png">
<h4>UserNameUserName</h4>
<h6>@screenName</h6>
</div>
<div id="info"></div>
<div id="text">
<p>asdfdafdaadd</p>
</div>
<div class="form-group">
<div class="col-xs-12">
<textarea class="form-control" rows="3"
 id="reply-text"></textarea>
</div>
<div class="col-xs-2 ">
<span class="help-block">140</span>
</div>
<div class="col-xs-5 col-xs-offset-5">
<button type="submit" id="reply-button"
class="btn btn-lg btn-primary pull-right">
<span class="glyphicon glyphicon-share-alt">
</span> Reply
</button>
</div></div></div>
<div id="conversation-item">
<h4>Original</h4>
<img src="star-half.png" alt="img">
<h5>userName</h5>
<h6>screenName</h6>
<div>text</div>
</div></div>
<!-- col-right END -->
</div><!-- modalbox END -->
<div id="tweet-send-area">
<div id="col-left">
<form class="form-horizontal">
<fieldset>
```

```
<legend id="title">Legend</legend>
<div class="form-group">
<label for="textArea"
class="col-xs-1 pull-left control-label">Tweet</label>
<div class="col-xs-12">
<textarea class="form-control" rows="2"
id="tweet-text"></textarea>
</div>
<div class="col-xs-2 ">
<span class="help-block">140</span>
</div>
<div class="col-xs-6 col-xs-offset-4">
<input type="button" class="btn btn-lg btn-primary pull-right"
id="tweet-button" value="Tweet"/>
</div></div>
</fieldset>
</form></div></div>
<div id="store"></div>
<script type="text/javascript">
<!--
app.initialize(index.initialize);
-->
</script>
</body>
</html>
```

A.1.7 tweet.js

```
/*
 * Copyright (C) The Language Grid Project.
 * All rights reserved.
 */
var index = function() {
var self = {
alertError : function(err) {
alert(err);
},
initialize : function(languageGrid, twitter) {
self.languageGrid = languageGrid;
self.twitter = twitter;
/**
 * 初期処理
 */
(function(){
//screenName取得
self.twitter.check().done(function(ret) {
```

```
$("#col-left #title").text("@"+ ret);
getTimeline(function(){
//タイムライン取得成功後の処理
interval();
});
}).fail(self.alertError);
})();
var modalBox = $('#modalbox');
 var layer = $('<div />').appendTo('body').addClass('layer');
var hashcount = 0;
var HASH_TAG = "";
 var targetLang = "en";
 var yourLang = "ja";

 /**
 * モーダルウィンドウを開く
 */
var openModal = function(){
//モーダルウィンドウのcss
 modalBox.css({
 top: ($(window).height() - modalBox.outerHeight()) / 2,
 left: ($(window).width() - modalBox.outerWidth()) / 2
});

 //表示する
$([modalBox[0],layer[0]]).fadeIn(50);

//レイヤークリックで閉じる
$(layer[0]).on("click", function(){
 $([modalBox[0],layer[0]]).fadeOut(50);
 });
}
$("#check").click(function(){
//init();
openModal();
});
$("#timelineUpdate").click(function(){
getTimeline(function(){
});
});
/**
 * タイムライン取得関数
 * f: タイムライン取得後に実行する関数
 */
function getTimeline(f){
```

```
self.twitter.getList(0,1000).done(function(ret) {
var timelineList = $("#timeline_list");
timelineList.empty();
var store = $("#store");
store.empty();
for (var i = 0; i < ret.length; i++) {
var item = ret[i];
var li = createListItemForData(item);
timelineList.prepend(li);
store.append(createPageItemForData(item));
}
f();
}).fail(self.alertError);
}
/**
 * col-center（真ん中のカラム）に表示されるツイートのリスト要素
 */
var createListItem = function(ret){
var id = ret.id;
var s = ret.status;
var mId = id; //status id
var imageURL = s.user.profileImageURL; //image URL
var userName = s.user.name; //user name
var text = s.text; // tweet text
var item =
'<li class="media" id="' + mId + '">
<a class="media-left pull-left" href="#">' +
'<img src="' + imageURL + ' " alt="..."></a>' +
 '<div class="media-body"><h4 class="media-heading">' +
userName + '</h4><div class="item-text">' + text +
 '</div></div></li>';
return item;
}
/**
 * ツイッターからのデータ取得時
 * col-right（右カラム）に表示されるツイートの詳細情報
 * 普段は見えない#storeという所に保存し、
 * 呼び出されると右カラムにコピーする
 */
var createPageItem = function(ret){
var id = ret.id;
var s = ret.status;
var mId = "page" + id;
var userName = s.user.name;
var screenName = s.user.screenName;
```

```
var text = s.text;
var imageURL = s.user.profileImageURL;
var replyId = ret.inReplyToStatusId;
var page = '<div id=' + mId + '><div id="user">
<img alt="img" src="' + imageURL + '">
<h4>' + userName + '</h4>
<h6>@' + screenName +
'</h6></div><div id="info"></div><div id="text">
<p>' + text + '</p></div>' +
'<div id="tweet-id">' + id + '</div>' +
'<div id="reply-id">' + replyId + '</div>' +
'<div id="image-url" class="">' + imageURL +'</div>' +
'<div class="form-group">' +
'<div class="col-md-12">
<textarea class="form-control" rows="5" id="reply-text">
</textarea></div>' +
'<div class="col-md-2 ">
<span class="help-block">129</span></div>' +
'<div class="col-md-5 col-md-offset-5">
<button type="submit" id="reply-button"
class="btn btn-lg btn-primary pull-right">' +
'<span class="glyphicon glyphicon-share-alt">
</span> Reply</button></div></div>' +
'</div>';
return page;
}
/**
 * データベースからのデータ取得時
 * col-center（真ん中のカラム）
 * に表示されるツイートのリスト要素
 */
var createListItemForData = function(ret){
var id = ret.statusId;
var mId = id; //status id
var imageURL = ret.imageURL; //image URL
var userName = ret.userName; //user name
var text = ret.text; // tweet text
var item =
'<li class="media" id="' + mId + '">
<a class="media-left pull-left" href="#">' +
'<img src="' + imageURL + ' " alt="..."></a>' +
 '<div class="media-body"><h4 class="media-heading">'
+ userName + '</h4><div class="item-text">' + text +
 '</div></div></li>';
return item;
```

```
}
/**
 * データベースからのデータ取得時
 * col-right（右カラム）に表示されるツイートの詳細情報
 * 普段は見えない#storeという所に保存し、
 *  呼び出されると右カラムにコピーする
 */
var createPageItemForData = function(ret){
var id = ret.statusId;
var mId = "page" + id; //status id
var imageURL = ret.imageURL; //image URL
var userName = ret.userName; //user name
var text = ret.text; // tweet text
var replyId = ret.inReplyToStatusId;
var screenName = ret.screenName;
var page = '<div id=' + mId + '><div id="user">
<img alt="img" src="' + imageURL + '">
<h4>' + userName + '</h4>
<h6>@' + screenName +
'</h6></div><div id="info"></div>
<div id="text"><p>' + text + '</p></div>' +
'<div id="tweet-id">' + id + '</div>' +
'<div id="reply-id">' + replyId + '</div>' +
'<div id="image-url" class="invisible">'
+ imageURL +'</div>' +
'<div class="form-group">' +
'<div class="col-xs-12">
<textarea class="form-control" rows="5" id="reply-text">
</textarea></div>' +
'<div class="col-xs-2 ">
<span class="help-block">129</span></div>' +
'<div class="col-xs-8 col-xs-offset-2">
<button type="submit" id="reply-button"
class="btn btn-lg btn-primary pull-right">' +
'<span class="glyphicon glyphicon-share-alt">
</span> Reply</button></div></div>' +
'</div>';
return page;
}
/**
 * 会話ツリーの末端　つまり一番最初のページを探して表示する
 */
function getConversationOrigin(page){
var id = page.find("#reply-id").text();
if(id < 0 ){
```

```
return page;
}else{
return getConversationOrigin($("#page" + id));
}
}
/**
 * 会話ツリーの一番根の文を表示するときに利用する
 */
function createConversationPage(page){
var imageURL = page.find("#image-url").text();
var userName = page.find("h4").text();
var screenName =page.find("h6").text();
var text = page.find("p").text();
var ret = '<h4>Original</h4>'+
'<img alt="img" src=' + imageURL + '>' +
'<h5>' + userName + '</h5>' +
'<h6>' + screenName + '</h6><div>' + text + '</div>';
return ret;
}
/**
 * リストのアイテムクリック時の処理
 * timelineのアイテムがクリックされたときの処理
 */
$("#timeline_list").on("click","li",function(){
var mId = "page" + $(this).attr("id");
var page = $("#" + mId).clone();
var str = $(this).find(".item-text").text();
var reg = new RegExp(" #collatran");
var reg2 = new RegExp($("body").find("legend").text());
str = str.replace(reg,"");
str = str.replace(reg2,"");
var screenName = page.find("h6").text();
if(screenName == $("#col-left legend").text()){
}else{
str = screenName + " " + str;
}
page.find(".help-block")
.text(140 - str.length - hashcount);
page.find("textarea").val(str);
$("#tweet-area").empty().append(page);
if(-1 == page.find("#reply-id").text()){
$("#conversation-item").empty();
$("#conversation-item").addClass("invisible");
}else{
var origin = getConversationOrigin(page);
```

```
$("#conversation-item").empty()
.append(createConversationPage(origin))
.removeClass("invisible");
}
openModal();
});
/**
 * ツイートボタンが押されたときの処理（左カラム
 * まず翻訳して
 * 同時に投稿する
 */
$("body").on("click","#tweet-button",function(){
var text = $("#col-left #tweet-text").val();
var len = 140 - text.length - hashcount;
if(len < 0){
alert("文字数オーバー(length over)");
return;
}
var lang =getYourLang();
var targetLang = getTargetLang();
self.languageGrid.translate(lang, targetLang, text)
.done(function(ret){
var ttext = ret;
var len = 140 - ttext.length - hashcount;
if(len < 0){
alert("翻訳の文字数オーバー(Translation length over)
原文を修正してください");
return;
}
if(text === ttext){
alert("original and translation are equal");
return;
}
self.twitter.collaborativeTweet(text +
 HASH_TAG ,ttext + HASH_TAG, lang , targetLang)
 .done(function(ret) {
$("#col-left #tweet-text").val("");
$("#col-right .help-block").text(140 - hashcount);
}).fail(self.alertError);
}).fail(self.alertError);
});
/**
 * replyボタンクリック時の処理
 */
$("#col-right").on("click","#reply-button",function(){
```

```
var id = $("#col-right").find("#tweet-id").text();
var reptext = $("#col-right").find("textarea").val();
var text = $("#col-right").find("#text p").text();
self.twitter.reply2(id,reptext + HASH_TAG,text)
.done(function(ret) {
 $([modalBox[0],layer[0]]).fadeOut(300);
}).fail(self.alertError);
});
/**
 * 言語選択ボタンが押されたときの処理
 */
$("#ja-button").click(function(){
yourLang = "ja";
targetLang = "en";
$(this).removeClass("btn-default")
.addClass("btn-primary");
$("#en-button").removeClass("btn-primary")
.addClass("btn-default");
$("#navtext").text("translate Japanese to English.");
});
$("#en-button").click(function(){
yourLang = "en";
targetLang = "ja";
$(this).removeClass("btn-default").addClass("btn-primary");
$("#ja-button").removeClass("btn-primary")
.addClass("btn-default");
$("#navtext").text("translate English to Japanese.");
});
/**
 * セレクターのyour language で選択している言語を取得し、
 *  valueの値を返す
 * 例：Japaneseが選択 -> ja
 * 入力される文章の言語
 */
var getYourLang = function(){
return yourLang;
}
/**
 * セレクターのtarget language で選択している言語を取得し、
 *  valueの値を返す
 * 例：Japaneseが選択 -> ja
 * この言語に翻訳される
 */
var getTargetLang = function(){
return targetLang;
```

```
}
/**
 * ツイートエリア
 * 現在入力されている文字数を表示する
 * 左カラムのinputフォーム
 */
$("#col-left")
.on("keyup","#tweet-text",function(){
var str = $(this).val();
$("#col-left span").text(140 - str.length - hashcount);
})
.on("paste","#tweet-text",function(){
setTimeout(function(){
var str = $(this).val();
$("#col-left span").text(140 - str.length- hashcount);
},5);
});
/**
 * リプライ
 * 現在入力されている文字数を表示する
 * 右カラムのinputフォーム
 */
$("#col-right")
.on("keyup","#reply-text",function(){
var str = $(this).val();
$("#col-right .help-block").text(140 - str.length - hashcount);
})
.on("paste","#reply-text",function(){
setTimeout(function(){
var str = $(this).val();
$("#col-right .help-block").text(140 - str.length- hashcount);
},5);
});
/**
 * 定期的に行なう処理
 * getListLength: 現在サーバーにあるstreamから取得
 */
var interval = function(){
self.twitter.getStream().done(function(ret) {
var timelineList = $("#timeline_list");
var store = $("#store");
for (var i = 0; i < ret.length; i++) {
var item = ret[i];
var li = createListItemForData(item);
timelineList.prepend(li);
```

```
store.append(createPageItemForData(item));
}
setTimeout(function(){
interval();
},5000);
}).fail(self.alertError);
}
}
};
return self;
}();
```

A.2 サーバーアプリケーション

A.2.1 TweetStore.java

```
package org.langrid.collatran.datastore.tweet;
import java.net.URISyntaxException;
import java.util.List;
import java.util.stream.Collectors;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import jp.go.nict.langrid.commons.beanutils.Converter;
import jp.go.nict.langrid.commons.rpc.intf.Parameter;
import org.eclipse.jdt.internal.compiler.ast.OR_OR_Expression;
import org.langrid.collatran.datastore.api.Tweet;
import org.langrid.collatran.datastore.api.TweetStoreService;
import org.langrid.collatran.datastore.bdict.Bdict;
import org.langrid.collatran.datastore.matching.MatchingStore;
import org.langrid.collatran.datastore.tweet.model.Tweet_;
import org.langrid.service.api.common.InvalidParameterException;
import org.langrid.service.api.common.ProcessFailedException;
public class TweetStore implements TweetStoreService{
@Override
public long getCount() {
return tweetOps.getCount();
}
@Override
public List<Tweet> list(
@Parameter(sample="0") int firstResult,
@Parameter(sample="10") int maxResults) {
return tweetOps.list(firstResult,
maxResults, Tweet_.tweetId)
.stream()
.map(v -> converter.convert(v, Tweet.class))
.collect(Collectors.toList());
```

```
}
@Override
public void clear() {
tweetOps.clear();
}
@Override
public void add(
Tweet tweet) {
tweetOps.add(converter.convert(tweet,
org.langrid.collatran.datastore.tweet.model.Tweet.class));
}
@Override
public void delete(String tweetId) {
tweetOps.delete(Tweet_.tweetId, Long.valueOf(tweetId));
}
private Converter converter = new Converter();
private static
Ops<org.langrid.collatran.datastore.tweet.model.Tweet>
tweetOps;
private static Ops<Bdict> bdictOps;
private static EntityManagerFactory emf;
static{
emf = Persistence.createEntityManagerFactory("DataStore");
tweetOps = Ops.of(emf.createEntityManager(),
 org.langrid.collatran.datastore.tweet.model.Tweet.class);
bdictOps = Ops.of(emf.createEntityManager(), Bdict.class);
}}
```

A.2.2 Servlet.java

```
package org.langrid.collatran.datastore.servlet;
import javax.servlet.annotation.WebServlet;
import jp.go.nict.langrid.servicecontainer
.handler.annotation.Service;
import jp.go.nict.langrid.servicecontainer
.handler.annotation.Services;
import jp.go.nict.langrid.servicecontainer.handler
.jsonrpc.servlet.JsonRpcServlet;
import org.langrid.collatran.datastore.bdict.BdictStore;
import org.langrid.collatran.datastore
.matching.MatchingStore;
import org.langrid.collatran.datastore.tweet.TweetStore;
@WebServlet(urlPatterns="/jsServices/*")
@SuppressWarnings("serial")
@Services({@Service(name="MatchingStore",
impl=MatchingStore.class)
```

```
,@Service(name="TweetStore", impl=TweetStore.class)
,@Service(name="BdictStore",impl=BdictStore.class)})
public class Servlet extends JsonRpcServlet {
}
```

A.2.3 EnEnAlignmentCreator.java

```
package org.langrid.collatran.datastore.matching;
import java.util.HashMap;
public class EnEnAlignmentCreator  {
public HashMap<String,String>
create(String text,String text2){
HashMap<String,String> map =
new HashMap<String,String>();
String atMark = "@[a-zA-Z0-9]*";
String sharp = "#[a-zA-Z0-9]*";
String mark = "[.]";
text = text.replaceAll(atMark, "")
.replaceAll(sharp, "").replaceAll(mark, "");
text2 = text2.replaceAll(atMark,"")
.replaceAll(sharp, "").replaceAll(mark, "");
text = text.trim() + " ExNxD";
text2 = text2.trim() + " ExNxD";
String[] texts = text.split("\\s\\s*");
String[] texts2 = text2.split("\\s\\s*");
int length1 = texts.length;
int length2 = texts2.length;
int index=0;
int index2 = 0;
for(int i=0;i<length1;i++){
if(texts2[index].equals(texts[i])){
++index;
if(index >= length2){
return map;}
continue;
}
for(int j=i;j<length1;j++){
if((index2 = this.getEqualWordIndex(texts[j],
texts2, index)) < 0){
continue;
}else{
map.put(this.mergeMorpheme(texts, i, j),
this.mergeMorpheme(texts2, index, index2));
i = j-1;
index = index2;
break;
```

```
}}}
return map;
}
/**
 *
 * @param word 検索する語
 * @param m 検索する文章
 * @param startIndex どこからmを検索するか
 * @return m内でwordと一致する語が含まれていた場合は
 * その場所を返し，無い場合は-1を返す
 *
 */
public int getEqualWordIndex(String word,
String[] m, int startIndex){
int size = m.length;
for(int i=startIndex;i<size;i++){
if(word.equals(m[i])){
return i;
}}
return -1;
}
/**
 *
 * @param m 文章を分割した文字列配列
 * @param start
 * @param end
 * @return m[start]からm[end]までの文字列を全て連結させた文字列
 */
public String mergeMorpheme(String[] m, int start, int end){
int size = m.length;
if(size <= end){
return null;
}
StringBuilder sb = new StringBuilder();
sb.append(m[start]);
for(int i=start+1;i<end;i++){
sb.append(" ");
sb.append(m[i]);
}
return sb.toString();
}
```

A.2.4 JaEnAlignmentCreator.java

```
package org.langrid.collatran.datastore.matching;
import java.net.URI;
```

```
import java.net.URISyntaxException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import jp.go.nict.langrid.service_1_2.AccessLimitExceededException;
import jp.go.nict.langrid.service_1_2.InvalidParameterException;
import jp.go.nict.langrid.service_1_2.
LanguagePairNotUniquelyDecidedException;
import jp.go.nict.langrid.service_1_2.NoAccessPermissionException;
import jp.go.nict.langrid.service_1_2.NoValidEndpointsException;
import jp.go.nict.langrid.service_1_2.ProcessFailedException;
import jp.go.nict.langrid.service_1_2.ServerBusyException;
import jp.go.nict.langrid.service_1_2.ServiceNotActiveException;
import jp.go.nict.langrid.service_1_2.ServiceNotFoundException;
import jp.go.nict.langrid.service_1_2.
UnsupportedLanguagePairException;
import jp.go.nict.langrid.service_1_2.
morphologicalanalysis.Morpheme;
import jp.go.nict.langrid.service_1_2.
translation.TranslationService;
public class JaEnAlignmentCreator {
public JaEnAlignmentCreator() {
// TODO Auto-generated constructor stub
}
public JaEnAlignmentCreator(TranslationService ts){
super();
this.ts = ts;
}
public HashMap<String,String> create(Morpheme[] m_ja,
String text_en,Morpheme[] m_en,String text_ja){
HashMap<String,String> map = new HashMap<String,String>();
String buffer="";
int count=0;
for(Morpheme m:m_ja){
try {
String word_ja = m.getWord();
String word_en = ts.translate("ja", "en", word_ja);
String word_en_temp= word_en.toLowerCase();
text_en = text_en.toLowerCase();
if(m.getPartOfSpeech().indexOf("noun") != -1){
buffer += m.getWord();
count++;
}else if(!buffer.equals("")){
if(count > 1){
String temp = ts.translate("ja", "en", buffer);
```

```
if(text_en.indexOf(temp.toLowerCase()) != -1)
map.put(buffer, temp);
}else{
String temp = ts.translate("ja", "en", buffer);
if(text_en.indexOf(temp.toLowerCase()) != -1)
map.put(buffer, temp);
}
count =0;
buffer = "";
}
} catch (AccessLimitExceededException
| InvalidParameterException
| NoAccessPermissionException | ProcessFailedException
| NoValidEndpointsException | ServerBusyException
| ServiceNotActiveException | ServiceNotFoundException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
return map;
}
private TranslationService ts;
}
```

A.2.5 History.java

```
package org.langrid.collatran.datastore.matching;
import java.util.ArrayList;
import java.util.List;
import org.langrid.collatran.datastore.api.MatchingEntity;
import com.sun.media.jai.opimage.AddCollectionCRIF;
public class History {
List<MatchingEntity> entities;
String from;
String to;
public History() {
// TODO Auto-generated constructor stub
entities = new ArrayList<MatchingEntity>();
}
public History(MatchingEntity entity){
super();
entities = new ArrayList<MatchingEntity>();
this.from = entity.getReplyTweetId();
this.to = entity.getTweetId();
add(entity);
}
```

```
public void add(MatchingEntity entity){
entities.add(entity);
}
public MatchingEntity getFromString(String word){
for(MatchingEntity entity:entities){
System.out.println(word + "+++" + entity.
getToWord().toLowerCase());
if(entity.getToWord().toLowerCase().
indexOf(word.toLowerCase()) != -1 ){
return entity;
}
}
return null;
}
public List<MatchingEntity> getEntities() {
return entities;
}
public void setEntities(List<MatchingEntity> entities) {
this.entities = entities;
}
public String getFrom() {
return from;
}
public void setFrom(String from) {
this.from = from;
}
public String getTo() {
return to;
}
public void setTo(String to) {
this.to = to;
}
}
```

A.2.6 MatchingStore.java

```
package org.langrid.collatran.datastore.matching;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.stream.Collectors;
import javax.mail.search.OrTerm;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
```

```
import org.eclipse.jdt.internal.compiler.ast.OR_OR_Expression;
import org.langrid.collatran.datastore.api.MatchingEntity;
import org.langrid.collatran.datastore.api.MatchingStoreService;
import org.langrid.collatran.datastore.api.Tweet;
import org.langrid.collatran.datastore.bdict.Bdict;
import org.langrid.collatran.datastore.bdict.Bdict_;
import org.langrid.collatran.datastore
.matching.model.MatchingEntity_;
import org.langrid.collatran.datastore.tweet.Ops;
import org.langrid.collatran.datastore.tweet.model.Tweet_;
import org.omg.CORBA.PUBLIC_MEMBER;
import jp.go.nict.langrid.commons.beanutils.Converter;
import jp.go.nict.langrid.commons.rpc.intf.Parameter;
import jp.go.nict.langrid.service_1_2.AccessLimitExceededException;
import jp.go.nict.langrid.service_1_2.InvalidParameterException;
import jp.go.nict.langrid.service_1_2.NoAccessPermissionException;
import jp.go.nict.langrid.service_1_2.NoValidEndpointsException;
import jp.go.nict.langrid.service_1_2.ProcessFailedException;
import jp.go.nict.langrid.service_1_2.ServerBusyException;
import jp.go.nict.langrid.service_1_2.ServiceNotActiveException;
import jp.go.nict.langrid.service_1_2.ServiceNotFoundException;
import jp.go.nict.langrid.service_1_2.translation.TranslationService;
import jp.go.nict.langrid.client.soap.SoapClientFactory;
import jp.go.nict.langrid.service_1_2
.morphologicalanalysis.MorphologicalAnalysisService;
public class MatchingStore implements MatchingStoreService {
public long getCount(){
return matchingOps.getCount();
}
@Override
public List<MatchingEntity> list(int firstResult, int maxResults){
return matchingOps.list(firstResult, maxResults
, MatchingEntity_.primaryId)
.stream()
.map(v -> converter.convert(v, MatchingEntity.class))
.collect(Collectors.toList());
}
@Override
public HashMap<String,String> getJaEnMap(String text_ja){
String text_en=null;
try {
text_en = tservice.translate("ja", "en", text_ja);
} catch (AccessLimitExceededException | InvalidParameterException
| NoAccessPermissionException | ProcessFailedException
| NoValidEndpointsException | ServerBusyException
```

```
| ServiceNotActiveException | ServiceNotFoundException e1) {
// TODO Auto-generated catch block
e1.printStackTrace();
return null;
}
try {
return jecreator.create(ma_ja.analyze("ja", text_ja),
text_en, ma_en.analyze("en", text_en), text_ja);
} catch (AccessLimitExceededException | InvalidParameterException
| NoAccessPermissionException | NoValidEndpointsException
| ProcessFailedException | ServerBusyException
| ServiceNotActiveException | ServiceNotFoundException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
return null;
}
public HashMap<String , String>
getJaEnMap(String text_ja, String text_en){
try {
return jecreator.create(ma_ja.analyze("ja", text_ja),
 text_en, ma_en.analyze("en", text_en), text_ja);
} catch (AccessLimitExceededException | InvalidParameterException
| NoAccessPermissionException | NoValidEndpointsException
| ProcessFailedException | ServerBusyException
| ServiceNotActiveException | ServiceNotFoundException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
return null;
}
@Override
public HashMap<String, String>
getEnEnMap(String text1, String text2){
return eecreator.create(text1, text2);
}
public void clear(){
matchingOps.clear();
}
public void add(MatchingEntity entity){
matchingOps.add(converter.convert(entity,
 org.langrid.collatran.datastore.matching
 .model.MatchingEntity.class));
}
@Override
```

```
public List<Tweet> getTweets(
@Parameter(sample="0") int firstResult,
@Parameter(sample="10") int maxResults) {
return tweetOps.list(firstResult, maxResults, Tweet_.tweetId)
.stream()
.map(v -> converter.convert(v, Tweet.class))
.collect(Collectors.toList());
}
public void addEnEn(Tweet t1, Tweet t2){
HashMap<String, String> map =
getEnEnMap(t1.getText(), t2.getText());
for(String key: map.keySet()){
this.add(new MatchingEntity(key, map.get(key),
"en", "en", "", t2.getStatusId(),
t2.getInReplyToStatusId(),t2.getScreenName(),""));
}
}
public void addJaEn(Tweet t_ja,Tweet t_en){
HashMap<String , String> map =
getJaEnMap(t_ja.getText(), t_en.getText());
for(String key: map.keySet()){
this.add(new MatchingEntity(key, map.get(key),
"ja", "en", "", t_en.getStatusId(),
t_en.getInReplyToStatusId(),t_en.getScreenName(),key));
}
}
@Override
public void match(String start,int endcount){
List<Tweet> list = getTweets(0,10000);
HashMap<String, Tweet> map =
new HashMap<String, Tweet>();
for(Tweet t: list){
map.put(t.getStatusId(),t);
}
long startid = Long.parseLong(start);
for(Tweet tweet:list){
long temp = Long.parseLong(tweet.getStatusId());
if(temp < startid){
continue;
}
if(tweet.getInReplyToStatusId().equals("-1")){
continue;
}
Tweet rtweet = map.get(tweet.getInReplyToStatusId());
if((rtweet.getInReplyToStatusId()).equals("-1")){
```

```
addJaEn(rtweet, tweet);
}else{
addEnEn(rtweet, tweet);
}
}
}
@Override
public void createDB(String id){
List<MatchingEntity> list = this.list(0, 1000);
List<MatchingEntity> tempList =
new ArrayList<MatchingEntity>();
HashMap<String, History> map =
new HashMap<String, History>();
for(MatchingEntity entity:list){
if(map.containsKey(entity.getTweetId())){
History history = map.get(entity.getTweetId());
history.add(entity);
map.put(entity.getTweetId(), history);
}else{
History history=new History(entity);
map.put(entity.getTweetId(), history);
}
}
for(MatchingEntity entity:map.get(id).getEntities()){
String fromString;
String toString = entity.getToWord();
String temp;
while(!entity.getFromLang().equals("ja") || entity == null){
temp = entity.getFromWord();
History history=map.get(entity.getReplyTweetId());
entity = history.getFromString(entity.getFromWord());
if(entity == null){
break;
}
if(entity.getFromLang().equals("ja")){
fromString = entity.getFromWord();
toString = entity.getToWord().toLowerCase()
.replaceAll(temp.toLowerCase(), toString);
break;
}
}
}
}
private Converter converter = new Converter();
private static Ops<org.langrid.collatran.datastore.matching
```

```
.model.MatchingEntity> matchingOps;
private static Ops<Bdict> bdictOps;
private static Ops<org.langrid.collatran.datastore.tweet
.model.Tweet> tweetOps;
private static EntityManagerFactory emf;
private static EnEnAlignmentCreator eecreator =
new EnEnAlignmentCreator();
private static JaEnAlignmentCreator jecreator;
static TranslationService tservice;
static MorphologicalAnalysisService ma_ja;
static MorphologicalAnalysisService ma_en;
private static String id = "";
private static String pass = "";
public static String jserver =
"http://langrid.org/service_manager/invoker/kyoto1
.langrid:KyotoUJServer";
public static String mecab =
"http://langrid.org/service_manager/invoker/kyoto1
.langrid:Mecab";
public static String url_tagger =
"http://langrid.org/service_manager/invoker/kyoto1
.langrid:Illinoispostagger";
static{
emf = Persistence.createEntityManagerFactory("DataStore");
matchingOps = Ops.of(emf.createEntityManager(),
org.langrid.collatran.datastore.matching.model.MatchingEntity.class);
tweetOps = Ops.of(emf.createEntityManager(),
org.langrid.collatran.datastore.tweet.model.Tweet.class);
bdictOps = Ops.of(emf.createEntityManager(), Bdict.class);
try {
tservice = new SoapClientFactory()
.create(TranslationService.class,
new URL(jserver), id, pass);
jecreator = new JaEnAlignmentCreator(tservice);
} catch (MalformedURLException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
try {
ma_ja = new SoapClientFactory().create(
MorphologicalAnalysisService.class, new URL(mecab), id,
pass);
} catch (MalformedURLException e) {
// TODO Auto-generated catch block
e.printStackTrace();
```

```
}
try {
ma_en = new SoapClientFactory().create(
MorphologicalAnalysisService.class,
new URL(url_tagger),
id, pass);
} catch (MalformedURLException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
}
```

A.2.7 BDictServlet.java

```
package org.langrid.collatran.datastore.servlet;
import javax.servlet.annotation.WebServlet;
import jp.go.nict.langrid.commons.util.function.Consumer;
import jp.go.nict.langrid.service_1_2
.bilingualdictionary.BilingualDictionaryWithLongestMatchSearchService;
import jp.go.nict.langrid.servicecontainer.handler.axis.SGAxisServlet;
import jp.go.nict.langrid.servicecontainer
.handler.jsonrpc.servlet.JsonRpcServlet;
import jp.go.nict.langrid.servicecontainer.handler
.loader.DefaultServiceFactoryLoader;
import jp.go.nict.langrid.servicecontainer.handler
.loader.ServiceFactoryLoader;
import jp.go.nict.langrid.servicecontainer
.handler.servlet.MultiHandlerServlet;
import jp.go.nict.langrid.serviceexecutor
.db.ConnectionParameters;
import jp.go.nict.langrid.serviceexecutor.db
.bilingualdictionary.BilingualDictionaryWithLongestMatchSearch;

@SuppressWarnings("serial")
@WebServlet(urlPatterns={"/bdictjs/*", "/bdictsoap/*"})
public class BDictServlet extends MultiHandlerServlet{
public BDictServlet() {
put("bdictjs", new JsonRpcServlet(){
@Override
protected ServiceFactoryLoader[] getDefaultServiceFactoryLoaders() {
return BDictServlet.this.getDefaultServiceFactoryLoaders();
}
});
put("bdictsoap", new SGAxisServlet(){
@Override
```

```
protected ServiceFactoryLoader[]
getDefaultServiceFactoryLoaders() {
return BDictServlet.this.getDefaultServiceFactoryLoaders();
}
});
}
protected ServiceFactoryLoader[]
getDefaultServiceFactoryLoaders() {
DefaultServiceFactoryLoader l = new DefaultServiceFactoryLoader();
l.put("TestBDict", BilingualDictionaryWithLongestMatchSearch.class,
new Class<?>[]
{BilingualDictionaryWithLongestMatchSearchService.class},
new Consumer<BilingualDictionaryWithLongestMatchSearch>() {
@Override
public void accept
(BilingualDictionaryWithLongestMatchSearch value) {
ConnectionParameters cp = new ConnectionParameters();
cp.setJndiDataSourceName("jdbc/CollatranDataStore");
cp.setDbDictionary("POSTGRESQL");
value.setConnectionParameters(cp);
value.setTableName("bdict");
value.setLanguageColumnNames("ja en");
}
});
return new ServiceFactoryLoader[]{l};
}
}
```

A.2.8 CompositeServiceServlet.java

```
package org.langrid.collatran.datastore.servlet;
import java.net.MalformedURLException;
import java.net.URL;
import javax.servlet.annotation.WebServlet;
import jp.go.nict.langrid.client.soap.SoapClientFactory;
import jp.go.nict.langrid.commons.util.function.Consumer;
import jp.go.nict.langrid.composite.translation
.TranslationCombinedWithBilingualDictionaryWithLongestMatchSearch;
import jp.go.nict.langrid.service_1_2.morphologicalanalysis
.MorphologicalAnalysisService;
import jp.go.nict.langrid.service_1_2.translation.TranslationService;
import jp.go.nict.langrid.service_1_2.translation
.TranslationWithTemporalDictionaryService;
import jp.go.nict.langrid.servicecontainer.handler.axis.SGAxisServlet;
import jp.go.nict.langrid.servicecontainer.handler
.jsonrpc.servlet.JsonRpcServlet;
```

```
import jp.go.nict.langrid.servicecontainer.handler
.loader.DefaultServiceFactoryLoader;
import jp.go.nict.langrid.servicecontainer.handler
.loader.ServiceFactoryLoader;
import jp.go.nict.langrid.servicecontainer.handler
.servlet.MultiHandlerServlet;
import jp.go.nict.langrid.servicecontainer.service
.ComponentServiceFactory;
import jp.go.nict.langrid.servicecontainer.service
.composite.AbstractCompositeService;
import jp.go.nict.langrid.serviceexecutor.db.ConnectionParameters;
import jp.go.nict.langrid.serviceexecutor.db.bilingualdictionary
.BilingualDictionaryWithLongestMatchSearch;

@SuppressWarnings("serial")
@WebServlet(urlPatterns={"/compositejs/*", "/compositesoap/*"})
public class CompositeServiceServlet extends MultiHandlerServlet{
public CompositeServiceServlet() {
put("compositejs", new JsonRpcServlet(){
@Override
protected ServiceFactoryLoader[]
 getDefaultServiceFactoryLoaders() {
return CompositeServiceServlet.this
.getDefaultServiceFactoryLoaders();
}
});
put("compositesoap", new SGAxisServlet(){
@Override
protected ServiceFactoryLoader[]
getDefaultServiceFactoryLoaders() {
return CompositeServiceServlet.this
.getDefaultServiceFactoryLoaders();
}
});
bdict = new BilingualDictionaryWithLongestMatchSearch();
ConnectionParameters cp = new ConnectionParameters();
cp.setJndiDataSourceName("jdbc/CollatranDataStore");
cp.setDbDictionary("POSTGRESQL");
bdict.setConnectionParameters(cp);
bdict.setTableName("bdict");
bdict.setLanguageColumnNames("ja en");
}
private BilingualDictionaryWithLongestMatchSearch bdict;

protected ServiceFactoryLoader[] getDefaultServiceFactoryLoaders() {
```

```
DefaultServiceFactoryLoader l = new DefaultServiceFactoryLoader();
l.put(
"TranslationCombinedWithBilingual
DictionaryWithLongestMatchSearch",
TranslationCombinedWithBilingual
DictionaryWithLongestMatchSearch.class,
new Class<?>[]
{TranslationWithTemporalDictionaryService.class},
new Consumer<AbstractCompositeService>() {
@Override
public void accept(AbstractCompositeService t) {
((AbstractCompositeService)t)
.setComponentServiceFactory(new ComponentServiceFactory(){
@SuppressWarnings("unchecked")
@Override
public <T> T getService(String invocationName
, Class<T> interfaceClass) {
if(invocationName.equals("MorphologicalAnalysisPL")){
return (T)getLangridService
(MorphologicalAnalysisService.class, "Mecab");
} else if(invocationName.equals
("BilingualDictionaryWithLongestMatchSearchPL")){
return (T)bdict;
} else if(invocationName.equals("TranslationPL")){
return (T)getLangridService(TranslationService.class
, "KyotoUJServer");
}
return null;
}
});
```