特別研究報告書

Wikipedia翻訳のための多言語議論の支援

指導教員　　石田　亨　教授

京都大学工学部情報学科

石田　憲幸

2010年2月3日

Wikipedia翻訳のための多言語議論の支援

石田　憲幸

内容梗概

今日では，Wikipediaが Web 上において有用なドキュメントになってきている．しかし，Wikipediaの記事数は言語によって大きく異なり，それに応じてWikipediaを活用できる度合いにも差が発生する．こういった言語による違いを埋めるためには，ある言語の記事を別の言語に翻訳することで，その記事に対応する別の言語の記事を作成する，ということが考えられる．しかし，翻訳が行われるときには問題が発生する．その 1 つは，あるコミュニティ独自の単語や言い回しの翻訳を行う場合に発生する問題で，その単語や言い回しに対応する概念がないコミュニティにおいて，対訳を見つけることができないというものである．こういった場合に対訳を見つけるためには議論を行う必要がある(翻訳のための議論)．

翻訳のための議論を行う場合には，ある単語や言い回しに対応する概念があるコミュニティ内の人と，それに対応する概念がないコミュニティ内の人との両方が議論に参加する必要がある．従って，翻訳のための議論は一般に多言語環境における議論になり，そこでは言語が大きな壁になってくる．この言語という壁に対応するためには，議論中において英語などの公用語を 1 つ決め，その言語で議論を行うことも考えられるが，そうした場合，その公用語が理解できない人は議論に参加できない．逆に，参加者が各々の母国語を用いることができれば，誰でも議論に参加できると言える．

参加者が各々の母国語を用いて議論に参加できるようにするためのツールとして，機械翻訳を利用することができるが，機械翻訳ではその精度における問題から，しばしば理解が困難な結果が得られる．さらに，翻訳のための議論を機械翻訳を介して行った場合，機械翻訳の精度とは異なる問題によって理解が困難な結果が得られることがある．それは主に，翻訳されるべきでない部分が翻訳されてしまう問題，複数回翻訳されてしまう問題，他の言語の語句が翻訳されない問題である．

本論文ではそれらを解決する方法として，当該部分を翻訳されないようにし，さらにその部分を説明する情報を付加するアルゴリズムを提案する．このアルゴリズムを用いると，翻訳のための議論を機械翻訳を介して行った場合に発生

する問題を解決することができる．また，多言語コミュニケーションを行うためのツールとして，多言語 BBS や多言語チャットなどがあるが，提案するアルゴリズムをサービス化することによって，そのツール内でサービス呼び出しをして，容易にアルゴリズムを適用した翻訳をすることができる．

さらに，実際にこのアルゴリズムを利用する場面として，MediaWiki で動作する拡張機能で，Wikipedia 上で多言語議論をするための機能を持っている Multilingual LiquidThreads を挙げ，Multilingual LiquidThreads にアルゴリズムを適用することによって，翻訳のための議論を行う場合に発生する問題を解決できることを示す．

Supporting Multilingual Discussion for Wikipedia Translation

Ishida Noriyuki

Abstract

In recent days, Wikipedia is coming to be a useful document in the Web. However, the number of Wikipedia's articles varies from language to language, in accord with which extent taking advantage of Wikipedia is different. An idea to eliminate the difference among languages is creating an article by translating the article. Yet, there are some problems in translation. One problem, which occurs when translating words or phrases specific in a community, is that we cannot find what to translate them in the community where there is no concept corresponding to the words or phrases. In order to find what to translate, it is neccesary to discuss (discussions about translation).

When we have discussions about translation, both participants in the community that has concepts, corresponding to the words or phrases and participants in the community that does not have. Therefore, discussions about translation are generally multilingual, there being the language barrier. A solution to tackle the barrier of language is using an official language such as English in discussions. However, this solution has a problem that those who cannot understand the official language cannot take part in the discussions. In contrast, if each participant can use his or her own mother language, anyone can take part in it.

Each participant can use his or her own mother language with machine translation, but because of its accuracy, machine translated sentences are often hard to understand. Moreover, when we have discussions about translation with machine translation, sentences can be hard to understand not because of its accuracy. That is mainly because of follows: to be translated section that should not be translated, to be translated multiple times, and to be held words untranslated if the words' language differs from the sentence.

In this paper, we propose an algorithm for their solution. The algorithm holds designated section untranslated, and attaches a description that describe the section. By using this algorithm, we can resolve the problems that occur

in discussions about translation with machine translation. In addition, we have implemented a web service for the proposed algorithm. Multilingual BBS or multilingual chat, multilingual communication tools can easily apply the algorithm, by calling the service.

Moreover, we describe Multilingual LiquidThreads as the situation this algorithm is used. Multilingual LiquidThreads is an extention for MediaWiki, which has a function to do multilingual discussion on Wikipedia. We show we can solve the problems that occur in discussions about translation, by applying algorithm to Multilingual LiquidThreads.

Wikipedia 翻訳のための多言語議論の支援

目次

第1章　はじめに

今日，Wikipediaが Web 上において有用なドキュメントになってきており，各記事が様々な場面で参照されるようになっている．しかし，Wikipediaの記事数は言語によって大きく異なり，英語のような記事数の多い言語では Wikipediaを活用できるが，記事数が少ない言語では，Wikipediaを十分に活用することができない．こうした言語による違いを埋めるために，Wikipediaでは記事の翻訳活動がすすめられている．ある言語の記事を別の言語に翻訳することで，その記事に対応する別の言語の記事を作成して記事数を増やすことができる．

記事の翻訳を行う際にはいくつかの問題が発生する．1つは，あるコミュニティ独自の単語や言い回しの翻訳を行う場合に発生する問題で，その単語や言い回しに対応する概念がないコミュニティにおいて，対訳を見つけることができないというものである．このほかにも，文化の違いによって記事の内容に対して検閲が行われたり，翻訳の際に編集者の解釈が加えられたりするという問題もあるが，この点についてはここでは触れない．

ある単語や言い回しに対応する概念がないコミュニティにおいて対訳を見つけるために，その概念があるコミュニティ内の人と議論を行って対訳を決定するという作業 (翻訳のための議論) がしばしば行われる．この議論には，その単語や言い回しに対応する概念がないコミュニティからの参加者と，その概念があるコミュニティからの参加者との両方が必要であるため，一般に多言語環境における議論になり，ここでは，言語が大きな壁となってくる．それに対応する1つの方法として，議論中の公用語を英語など1つの言語に決めることが考えられるが，その公用語が理解できない人は議論に参加することができない．逆に，参加者が各々の母国語を用いることができれば，誰でも議論に参加できると言える．

参加者が各々の母国語を用いるようになると，その言語を別の参加者の母国語に変換する作業が必要になる．こうした作業を行うツールとして機械翻訳が利用できる．しかし，機械翻訳はその精度における問題から，しばしば理解が困難な結果が得られてしまう．さらに，翻訳のための議論を行った場合には，機械翻訳の精度による問題とは別の問題が発生する．

機械翻訳を用いて翻訳のための議論を行った場合に発生する問題として，翻訳されるべきでない部分が翻訳されてしまう問題がある．これは，翻訳される

ことによって文の意図が変わってしまったり，入力者が文の一部に対して翻訳されないことを望む場合があるために生じる．また，別の問題として，複数回の翻訳によって文の理解が困難になる問題がある．これは，議論の性質上，引用や返信が多用されるために生じる．何度も引用・返信される文においては，機械翻訳結果に対して引用・返信されることもあるが，そうなると，機械翻訳結果が再び機械翻訳されることになる．このように何度も翻訳されると，機械翻訳の精度が不十分であるために，文の理解が困難になってしまう．

以下では，ここに述べたような問題を解決するようなアルゴリズムを提案する．また，それを実現するプログラムを実装・評価し，実際に問題を解決できていることを示す．

第2章　研究の背景

今日，Wikipedia が Web 上において有用なドキュメントになってきている．Wikipedia は，オンライン上で百科事典を作ることを目的として運営されているコンテンツで，様々な場面で参照されるようになってきている．Wikipedia は世界各国の言語で展開されており，だれもが自由に記事の編集をすることができる．しかし，言語により記事の編集活動の活発さに差があり，そのため記事数にも差が発生している．例えば，英語では約 320 万記事，ドイツ語では約 99 万記事ある一方，記事数としては 0 である言語もある[1)]．こうした違いを埋めるために，ある記事を翻訳して別言語の記事とする活動が進められている．記事にはしばしばコミュニティ独自の単語や言い回しが含まれ，それの対訳を決めるための議論が行われる (翻訳のための議論)．

翻訳のための議論を行う場合には，ある単語や言い回しに対応する概念があるコミュニティ内の人と，それに対応する概念がないコミュニティ内の人との両方が議論に参加する必要がある．従って，翻訳のための議論は一般に多言語環境における議論になり，そこでは言語が大きな壁になってくる．この言語という壁に対応するためには，議論中において英語などの公用語を 1 つ決め，その言語で議論を行うことも考えられるが，そうした場合，その公用語が理解できない人は議論に参加できない．逆に，参加者が各々の母国語を用いることができれば，誰でも議論に参加できる．

[1)] `http://meta.wikimedia.org/wiki/List_of_Wikipedias` (List of Wikipedias)

こういった多言語環境においてコミュニケーションするためのツールとして，多言語チャットや多言語 BBS などのツールが利用できる．AmiKai が開発した AmiChat は，多言語チャットシステムで，ユーザが入力した文が自動的に他の言語に機械翻訳され，異なる母国語の人でも文の意味が理解できるようになっている [1]．また，多言語コラボレーションを行うための基盤として言語グリッドというものがあり [2]，これを利用した多言語 BBS などのツールが，Language Grid Toolbox 上で公開されている．

このように，多言語コミュニケーションツールはいくつか開発されているが，翻訳のための議論を行う場合に発生する問題については考慮していない．従って，以下では翻訳のための議論を行う場合に発生する問題を解決する方法について考えていく．

第 3 章　翻訳のための議論

3.1　概要

翻訳のための議論を行う場合には，いくつかの問題が発生する．ここでは主な問題として，翻訳されるべきでない部分が翻訳される問題と，複数回翻訳される問題を挙げ，整理していく．

3.2　翻訳されるべきでない部分が翻訳されてしまう問題

翻訳のための議論を行う場合の問題の 1 つは，翻訳されるべきでない部分が翻訳されてしまう問題である．これは，議論中において表記を尋ねる場合に発生する (図 3.1)．

“札幌”は英語で “Sapporo”と書かれますか？
⇒ Is “Sapporo” written with “Sapporo” in English?

図 3.1: 翻訳されるべきでない部分の存在

この場合，原文では「札幌」が「Sapporo」という表記であるかを尋ねているが，機械翻訳によってその意図が変わってしまっており,「札幌」という部分は翻訳されるべきでない部分であると考えることができる．

また，文の意図が変わらなくても，入力者が翻訳されないことを望む場合に

は，翻訳されるべきでない部分が翻訳されてしまうことになる (図 3.2).

宮崎駿は、“それから海の魚なのに平気で水道の水の中に入れてるんですよ。普通、水道の水に入れたら死んじゃいますよ、金魚だって。”と言っています。
⇒ Shun Miyazaki says, “And though it’s fish from sea, it’s kept in the water of waterworks calmly. If it’s usually put in water of waterworks, I die, it’s a goldfish.”

図 3.2: 入力者が翻訳されないことを望む場合

この例では，引用符内について翻訳されないことを入力者が望んでいる可能性がある.

このような，翻訳されるべきでない部分が翻訳されてしまう問題の解決策としては，翻訳されるべきでない部分を翻訳しないようにするということが考えられるが，それだけでこれらの問題に対処することはできない．実際，図 3.1 の場合は対応できるが，図 3.2 の場合は，翻訳しないようにした時に，日本語を理解できない人に引用符内の意味が伝わらなくなってしまう.

3.3 複数回翻訳されてしまう問題

議論を行う際には，過去に投稿・発言された内容を引用し，その部分に返信する形にする場合がよくある．これが多言語で行われると，引用されるたびにその部分が何度も翻訳され，そのたびに理解が困難な結果になっていく (図 3.3).

図 3.3 の場合においても，原文を複数回機械翻訳したために，結果の文の意味が理解困難になってしまっている．また，原文と大きく変化してしまっているため，同じ文を引用しているということが分からなくなってしまう可能性がある.

宮崎:実は、最初はカエルでいこうかと思ったんです。 でも、カエルのキャラクターは使われすぎてて、いくらデザインを変えてもピンと来ない。
⇒ Miyazaki: I thought actually, the beginning went by a frog. But the character of the frog is used too much, and even if the design is changed, how much don't you come tightly?
⇒宮崎： 私は、実際、最初がカエルによって過ぎたと思った。 しかし、カエルのキャラクターはあまりにも多く使われて、デザインが変更されても、どのくらい堅く来るわけではないか？

図 3.3: 複数回の翻訳により理解が困難になる例

3.4 他の言語の語句が翻訳されない問題

翻訳のための議論においては，ある言語の文の中に別の言語の語句が含まれることがよくある．その際に，その部分の言語がわからないと，翻訳することができない．以下の図 3.4 は，英語の語句を含む日本語の文をドイツ語に翻訳した例である．

“ドア”は英語で “Door”と書かれる．
⇒ “Tür” wird als “Door” in Englisch geschrieben.

図 3.4: 文中に別の言語の語句が含まれる場合

この例では，“Door”という英語の部分が，他の部分同様日本語であると解釈されてしまい，翻訳されずにそのまま出力されてしまっている．その部分が，翻訳されるべきである部分かどうかという問題はあるが，翻訳されないままでは，英語を理解できない人には伝わらなくなってしまう．

第 4 章 「翻訳のための議論」の翻訳

4.1 概要

前章では，翻訳のための議論を行う場合に発生する問題について整理した．以下では，その問題を解決するためのアルゴリズムを提案する．

4.2 問題の解決手法

4.2.1 翻訳されるべきでない部分が翻訳されてしまう問題

この問題を解決するためには，翻訳されるべきでない部分を翻訳しないようにすること (翻訳の回避) が考えられる．しかし，3.2 節で述べたように，それだけでは不十分である．ここではそれを解決するために，その部分の機械翻訳を説明文として付加することで解決を図る．

4.2.2 複数回翻訳されてしまう問題

この問題は，一度翻訳された文が再び翻訳されるために発生する．従ってこれを解決するためには，それを防げばよい．具体的な手法としては，引用などにおいて原文を保持しておき，翻訳される際には，原文と置き換えて翻訳するようにすればよい．原文を保持するには，翻訳の回避が利用できる．

4.2.3 他の言語の語句が翻訳されない問題

この問題は，別の言語の語句を特定する手段がないために発生する．従ってこれを解決するためには，その語句の部分を特定できるようにすればよい．具体的な手法としては，別の言語の語句をタグで囲むことで解決する．このとき，実際に翻訳するには，その部分の言語を特定する必要があるので，タグに言語コードを情報として付加する．

4.3 アルゴリズムの要件

4.2 節で述べた解決手法に基づき，問題を解決するためのアルゴリズムは以下の要件を満たす必要がある．

1. 翻訳の回避を示すタグで囲まれた部分については，翻訳の回避を行う．
2. 翻訳の回避が行われている部分には，その部分の翻訳結果を説明文として与える．
3. 翻訳を行う際には，説明文の部分を削除してから翻訳する．
4. 言語を表すタグで囲まれた部分については，その言語を翻訳元言語として翻訳を行う．

4.4 アルゴリズムの動作方針

解決のためのアルゴリズムは以下の方針で構成される．

1. 入力は原文 α，翻訳元言語 X，翻訳先言語 Y である．

2. α は言語 X の文であり，アルゴリズムは α を言語 Y に翻訳する．
3. α には，翻訳の回避を行う部分の集合が明示されている．
4. α の翻訳する部分は，言語 X から言語 Y の機械翻訳で結果を得る．
5. α の翻訳を回避する部分は翻訳せず，その部分に対する説明文を結合して結果とする．
6. 説明文は，翻訳を回避した部分の翻訳とし，その中に翻訳を回避する部分がさらに含まれていれば，その部分に対して翻訳の回避を行い，説明文を与える．
7. 説明文にその部分の言語 Z が指定されていれば，その翻訳は言語 Z から言語 Y，指定されていなければ，言語 X から言語 Y で行う．
8. 過去のアルゴリズム適用によって文中に説明文が既に含まれている場合，それらを除去してからアルゴリズムを適用する．

4.5 アルゴリズム

解決のためのアルゴリズムは次の Algorithm1 のようになる．

Algorithm 1 $MetaTranslate(\alpha)$

α：原文

X：原文の言語

Y：翻訳先の言語

$Replace(\alpha, A, B)$：文字列 α 中の部分文字列 A を文字列 B に置換する関数

$Translate(\beta, X, Y)$：文字列 β を言語 X から言語 Y に翻訳する関数

[：翻訳の回避を行う部分の開始を示す記号

]：翻訳の回避を行う部分の終了を示す記号

n：翻訳の回避を行う部分の数

S_i $(1 \leq i \leq n+1)$：翻訳の回避を行わない部分の文字列

T_i $(1 \leq i \leq n)$：翻訳の回避を行う部分の文字列

U_i $(1 \leq i \leq n)$：T_i に対応する中間コード (機械翻訳の影響を受けない)

D_i $(1 \leq i \leq n)$：T_i に対応する説明文

Require: $\alpha = S_1 + [+ T_1 +] + S_2 + [+ T_2 +] + ... + [+ T_n +] + S_{n+1}$

for $i = 1$ to n **do**
  $\alpha \leftarrow Replace(\alpha, T_i, U_i)$
end for
$\alpha \leftarrow Translate(\alpha, X, Y)$
for $i = 1$ to n **do**
  $D_i \leftarrow MetaTranslate(T_i)$
  $\alpha \leftarrow Replace(\alpha, U_i, T_i + \text{“(”} + D_i + \text{“)”})$
end for
return α

また，アルゴリズムの動作フローは図 4.1 のようになる．

4.6 入れ子について

4.5 節のアルゴリズムでは，翻訳を回避する部分の入れ子構造を考慮している．これは，翻訳を回避する部分の中に翻訳を回避する部分が含まれる場合があることを考慮している．例えば，引用の部分については，4.2 節で述べたように，翻訳の回避を用いて複数回の翻訳を防ぐことにしている．したがって，図 4.2 のような引用する部分に翻訳を回避する部分が含まれる場合にも対応する必要がある．

この例の場合，引用文の中の「札幌」という部分は翻訳されるべきではない．こういった場合に対応するためには，入れ子に対応する必要がある．

4.7 言語の指定について

4.5 節のアルゴリズムでは，言語を指定するとそれを翻訳元言語として翻訳することができるようになっている．これは，説明文を付加する際に特に重要になってくる．説明文を付加するには，その部分を機械翻訳する必要があるため，翻訳元言語が指定されていると適切な結果が得られると考えられる．

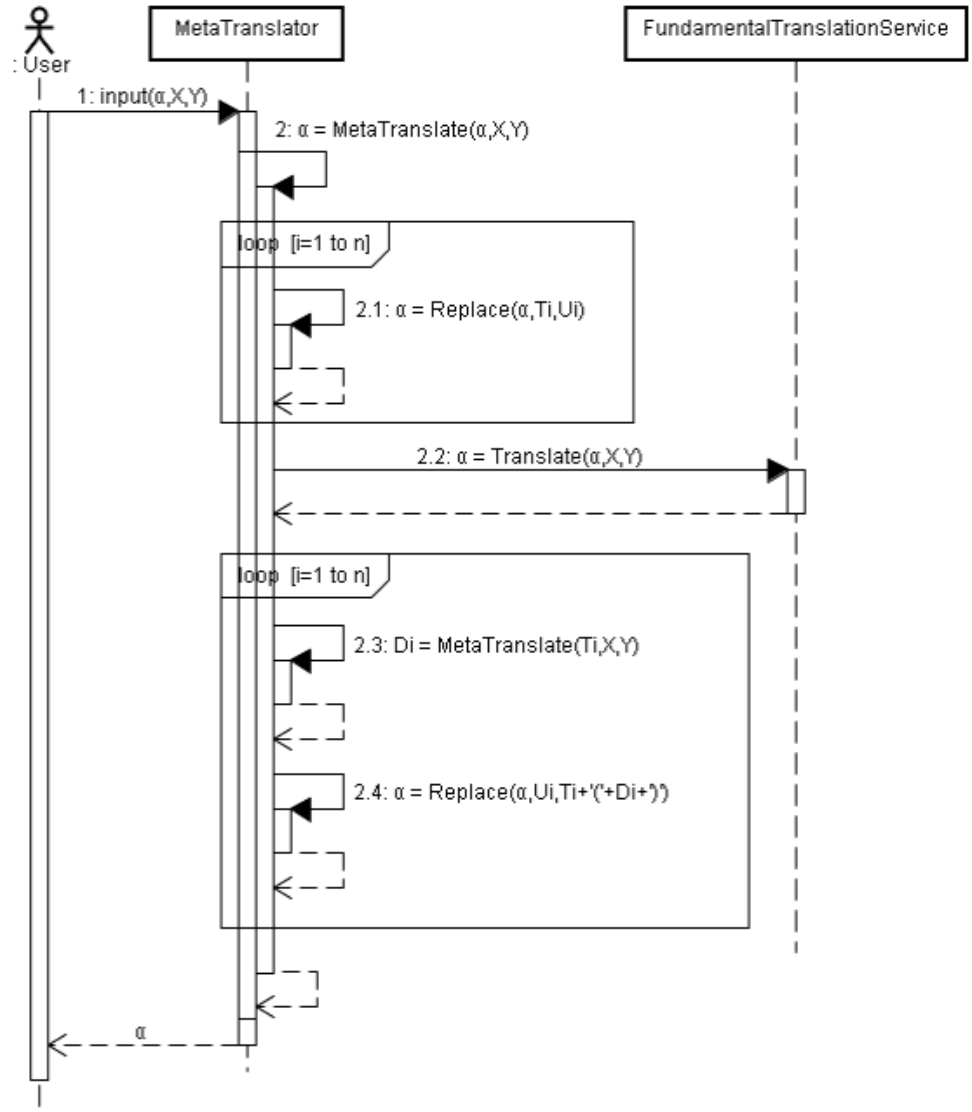

図 4.1: アルゴリズムの動作フロー

> “札幌”は英語で “Sapporo”と書かれますか？
はい.
⇒ > Is “Sapporo” written with “Sapporo” in English?
Yes.

図 4.2: 入れ子が必要となる場合

第5章　MediaWiki上での実装

5.1　概要

以下では，前章で提案したアルゴリズムをMediaWiki上で実装する際のシステム構成や仕様，また，使用例について述べる．

5.2　MediaWikiについて

5.2.1　MediaWikiとLiquidThreads

MediaWikiは，Wikipediaのために書かれたフリーソフトウェアパッケージ[1)]で，さまざまな拡張機能が開発されている．その中の拡張機能の1つに，LiquidThreadsという拡張機能がある．これは，Wikipedia上で議論を行う機能を提供している．LiquidThreadsを利用すると，Wikipediaの各記事ごとに議論ページをもたせることができ，その中でその記事に対する議論を行うことができる．

5.2.2　Multilingual LiquidThreads

LiquidThreadsには多言語議論を行う機能はないが，現在，LiquidThreadsを多言語議論で利用するための拡張機能，Multilingual LiquidThreadsが開発されている．この拡張機能を用いれば，Wikipediaの記事の翻訳を行うための多言語議論を行う環境を提供することができる．以下では，前章で提案したアルゴリズムを，このMultilingual LiquidThreadsを拡張する形で組み込むことを考える．

5.3　組み込むプログラムについて

5.3.1　概要

今回実装したプログラムは，第4章のアルゴリズムを実現するものである．以下では，そのプログラムの仕様について述べる．

5.3.2　翻訳の回避の実装

翻訳の回避の実装については，翻訳を回避する部分を一度別の文字列で置き換え，機械翻訳の後に再度元の文字列に置き換え直すという方法で行っている．このとき，置き換える文字列は，機械翻訳の前後で変化しない文字列(中間コード)を用いている．この方法は，辞書連携翻訳を行う際にも用いられている手法である．流れとしては，翻訳する前に，中間コードで置き換え，翻訳してから

---

1) http://www.mediawiki.org/wiki/MediaWiki/(MediaWiki)

元の文字列に置き換え直すようになっている．第 4 章のアルゴリズムでは，さらに説明文を付与する必要があるが，翻訳後に置き換え直す際に，置き換え後の部分に対して説明文を与えることで実現できる．

5.3.3 動作のモード

今回のプログラムは，引用符で囲まれた部分を翻訳の回避をするだけの機能を持った簡易版 (relaxed mode) と，[tag] と [/tag] の間に囲むことによって，入れ子にも対応した翻訳の回避をする厳密版 (strict mode) をもっている．

翻訳の回避をする対象となると考えられる部分は，あらかじめ引用符などで囲み，文中のそれ以外の部分と差別化されている場合が多い．relaxed mode は，そういった文章に対しては，手を加えなくてもアルゴリズムを適用するだけで，翻訳の回避をすることができる．しかし，引用符で翻訳の回避を行う部分の始まりと終わりを区別することは困難であるため，入れ子に対応することも困難となる．図 4.2 で示したような，引用された部分の中でさらに翻訳の回避が必要となる場面では，入れ子が有効に利用できるため，strict mode で入れ子に対応している．

5.3.4 言語指定機能

翻訳を回避するタグで囲まれた部分については，< 言語コード ></言語コード > で囲むことで，言語を指定することができる．

5.4 Web サービス化

5.4.1 Web サービス化について

今回のプログラムは，多言語議論を行うことのできる多言語 BBS や多言語チャットのようなツールの中に組み込み，そこから利用するという形が主になる．従って，そういったツールに容易に組み込めるように，柔軟かつ単純な仕様をもった Web サービスとして実装することが望ましいと考えられる．

以下では，実際に Web サービス化したプログラムの仕様について述べる．

5.4.2 Web サービスとしての仕様

今回のプログラムは，図 5.1 の MetaTranslatorService 部分のように，翻訳サービスの入力とする文を変形させ，さらに，翻訳サービスの出力となる文を変形させて結果を得るものである．

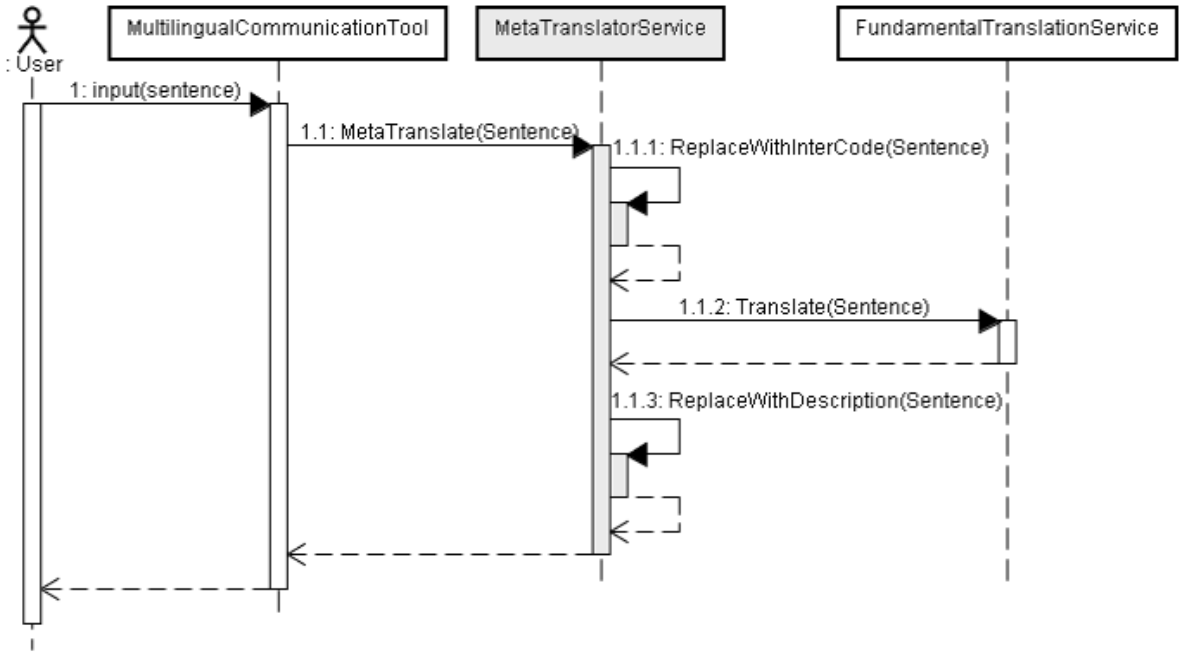

図 5.1: サービスの動作概要

今回実装したサービスのインターフェース仕様は次に示す (図 5.2).

入力:	sourceLang	翻訳元言語コード
	targetLang	翻訳先言語コード
	source	入力文
	mode	サービスの動作モード (relaxed または strict を指定)
	soap_options	SOAP 呼び出しのオプション
	service_wsdl	翻訳サービスの WSDL
	service_function	翻訳サービスにある翻訳メソッド名
	service_parameters	翻訳メソッドの呼び出しに必要なパラメータ
出力:	string	翻訳結果

図 5.2: MetaTranslator のインターフェース仕様

実装したMetaTranslatorServiceは，図5.2の仕様に基づいてSOAPから呼び出すことが可能である (図5.3, 図5.4).

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.(
  <soapenv:Header/>
  <soapenv:Body>
    <met:meta soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <sourceLang xsi:type="xsd:string">ja</sourceLang>
      <targetLang xsi:type="xsd:string">en</targetLang>
      <source xsi:type="xsd:string">[tag]生業[/tag]は[tag]Bread-and-butter job[/tag]に翻訳される・</source>
      <mode xsi:type="xsd:string">strict</mode>
      <soap_options xsi:type="xsd:string">user:aaaaa;
                                         pass:*****;
                                         encoding:UTF-8;
                                         timeout:15;</soap_options>
      <service_wsdl xsi:type="xsd:string">http://xxx.xxx/Translator?wsdl</service_wsdl>
      <service_function xsi:type="xsd:string">translate</service_function>
      <service_parameters xsi:type="xsd:string">sourceLang:$sourceLang;
                                               targetLang:$targetLang;
                                               source:$source;</service_parameters>
    </met:meta>
    </soapenv:Body>
</soapenv:Envelope>
```

図5.3: SOAPメッセージ例

```
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-ENV="http://schemas.
  <SOAP-ENV:Body>
    <ns4:metaResponse>
      <translateReturn xsi:type="xsd:string">[tag]生業(Bread-and-butter job)[/tag] is translated into [tag]Bread-and-butter job[/tag].</
    </ns4:metaResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

図5.4: SOAPメッセージ応答例

5.5 MediaWikiのシステム構成

5.5.1 概要

MediaWikiへのサービス組み込みについて考える前に，MediaWikiのシステム構成について説明する．MediaWikiのシステム構成は図5.5に示すようなものになっている．

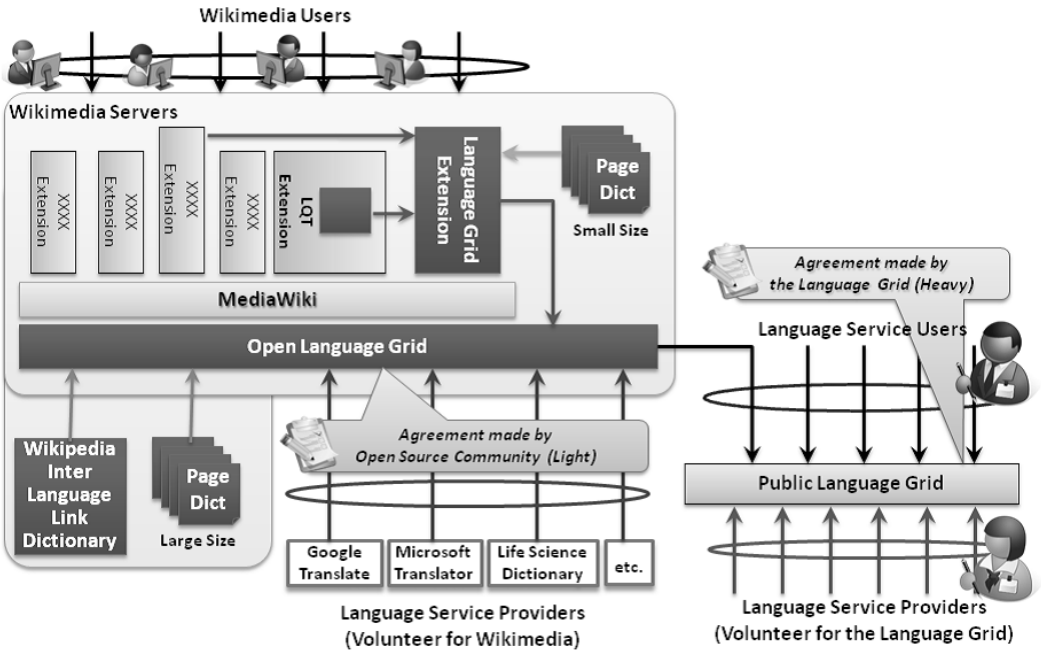

図 5.5: MediaWiki のシステム構成
(出典:http://www.langrid.org/mwikidev/demo_en/index.php/Main_Page)

MediaWiki は，図のように拡張機能 (Extension) をインストールすることで，様々な機能を与えることができる．多言語議論は，上で述べられている Multilingual LiquidThreads を拡張機能としてインストールすることで実現できる．Multilingual LiquidThreads Extension は図 5.5 の “LQT Extension” に対応している．

Multilingual LiquidThreads Extension を動作させるのには，図のように “Language Grid Extension” が必要となる．この拡張機能は，MediaWiki 上に言語グリッドを利用した多言語環境を構築するためのものである．Multilingual LiquidThreads で議論の翻訳を行うには，“Language Grid Extension” の翻訳メソッドを呼び出す．こうして，“Language Grid Extension” を通して言語グリッド基盤上での翻訳が行われる．このため，言語グリッドがサポートしている辞書連携翻訳なども行うことができる．

5.5.2 MediaWiki へのサービス組み込み

上で述べたシステム構成に基づき，今回実装した MetaTranslatorService の組み込みについて考える．MetaTranslatorService は，翻訳する文に作用するサービスなので，翻訳メソッドを呼び出す前後で作用するのが適切であると考えら

れる (図 5.6).

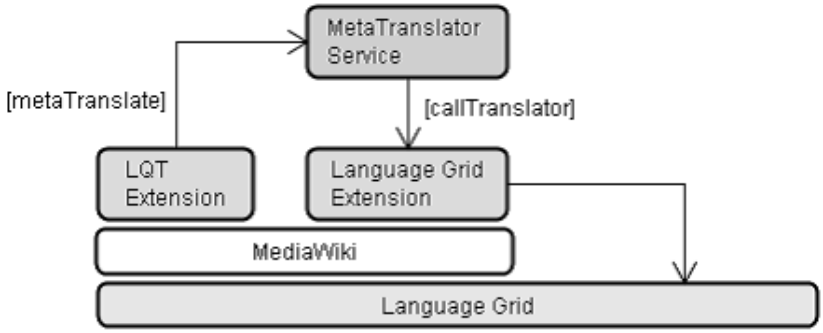

図 5.6: 主要部分のモジュール構成

MetaTranslatorService は，図 5.6 のように Multilingual LiquidThreads Extension から呼び出す．そして，MetaTranslatorService の中で Language Grid Extension を通じて翻訳サービスを呼び出し，アルゴリズムを実現する．翻訳の呼び出しなどの主要な部分のパッケージ図を図 5.7 に示す.

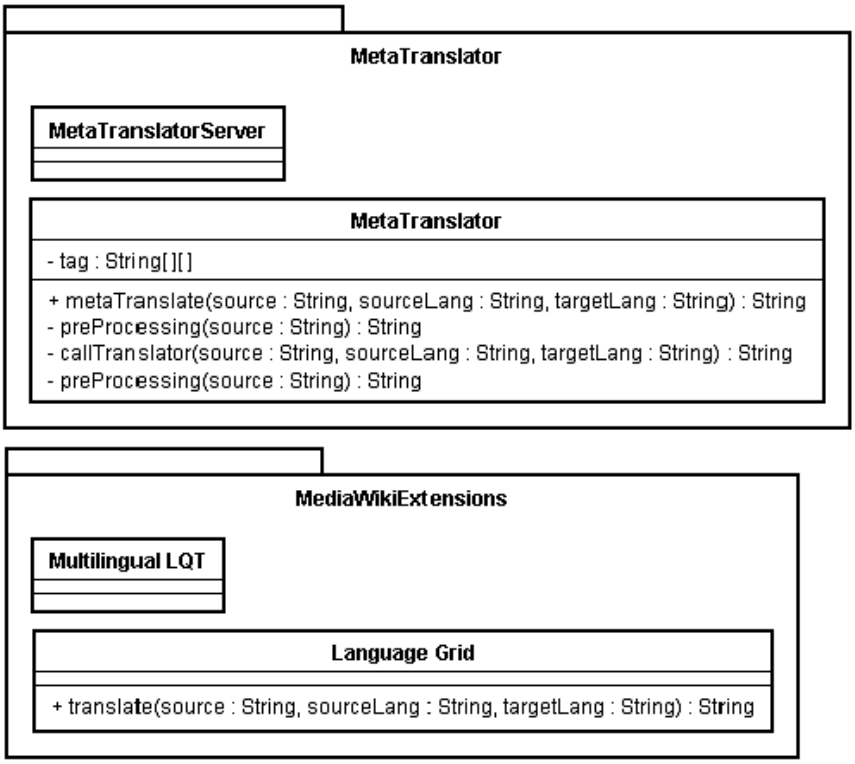

図 5.7: 主要部分のパッケージ図

現在の Multilingual LiquidThreads では，翻訳の際に Language Grid の translate メソッドを呼び出しているが，その代わりに，MetaTranslator の metaTranslate メソッドを呼び出すと，アルゴリズムが適用できる．metaTranslate メソッドの中では，preProcessing, callTranslator, postProcessing の 3 つのメソッドがこの順で実行されるが，callTranslator メソッドでは Language Grid の translate メソッドを呼び出す．

5.5.3 辞書

MediaWiki 上では，言語グリッドを利用して辞書連携翻訳を行うことができる．図 5.5 で見ると，“Wikipedia Interlanguage Link Dictionary” や “Page Dict” が辞書にあたる．“Wikipedia Interlanguage Link Dictionary” は，Wikipedia の記事から抽出した辞書で，Wikipedia のコミュニティ内で有効利用できると考えられる．“Page Dict” も Wikipedia のコミュニティ内で有効利用できる辞書であるが，ここでは詳しく触れない．以下では，“Wikipedia Interlanguage Link Dictionary” について述べる．

Wikipedia の記事には，言語間リンクというものが張られている．これは，ある言語の記事と別の言語の記事を対応づけるものである (図 5.8)．

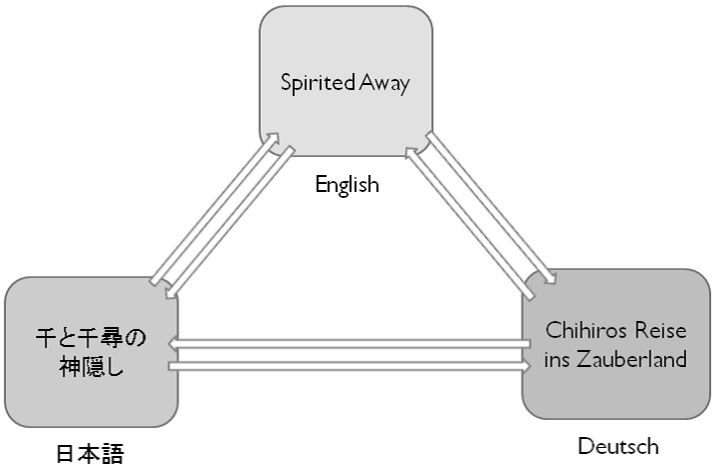

図 5.8: 言語間リンク

この言語間リンクを用いて，タイトル名で対応づけて辞書にしたものが，“Wikipedia Interlanguage Link Dictionary” である．この辞書は，Wikipedia の性質上，非常に大きなエントリを持っている．実際，英語版 Wikipedia の約 320 万記事

から抽出すると，約 150 万の辞書エントリが抽出できた．

Wikipedia から抽出したエントリは，全てが利用できるわけではない．例えば，“Category:Agriculture”↔“Category:農業”といった，存在していても実際には利用されることのないエントリがある．これは，辞書のエントリが完全一致で検索されるためである．つまり，このエントリに適合するためには “Category:” の部分まで入力する必要があるが，そういった入力が行われることはないと考えられる．

また，さらに問題のあるエントリとして，“Genkan”↔“玄関”といった，存在することで逆に悪影響を与えるエントリがある．これは，Wikipedia Interlanguage Link Dictionary は，翻訳器が持っている辞書よりも優先度が高い Global 辞書として登録されているため，もし翻訳器に “玄関” の対訳が存在していても “Genkan” で置き換えられてしまうためである．また，Wikipedia の大多数のエントリは大文字で始まるため，固有名詞でないエントリを持っていると，それらがすべて大文字で始まる文字列で置き換えられてしまうという問題もある．

このことから，抽出したエントリはフィルタリングで除去する必要がある．特に，上で述べた例のうち後者の問題は，fail-safe 性の保証のためにも対応が必要である．エントリ数を減らすことで，辞書連携翻訳の高速化を図ることもできる．

5.6 MediaWiki での使用例

実際に MediaWiki に組み込んで今回のアルゴリズムを動作させた例を以下に示す (図 5.9, 図 5.10)．

第 6 章 評価

6.1 概要

以下では，今回実装したプログラムが実際に第 3 章で述べた問題を解決するのに有効であることを評価実験により示す．

6.2 評価実験

アルゴリズムおよびプログラムの有効性を確かめるために，評価実験を行った．今回の実験は，Wikipedia 上の記事で行われる議論を想定して，通常の機械

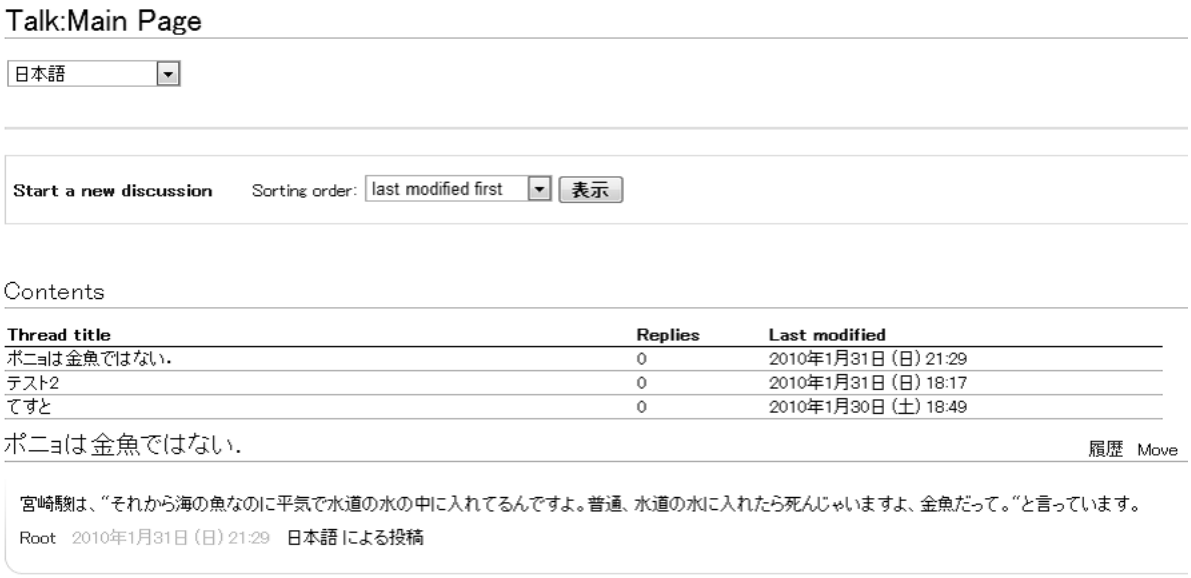

図 5.9: MediaWiki での動作例 (日本語)

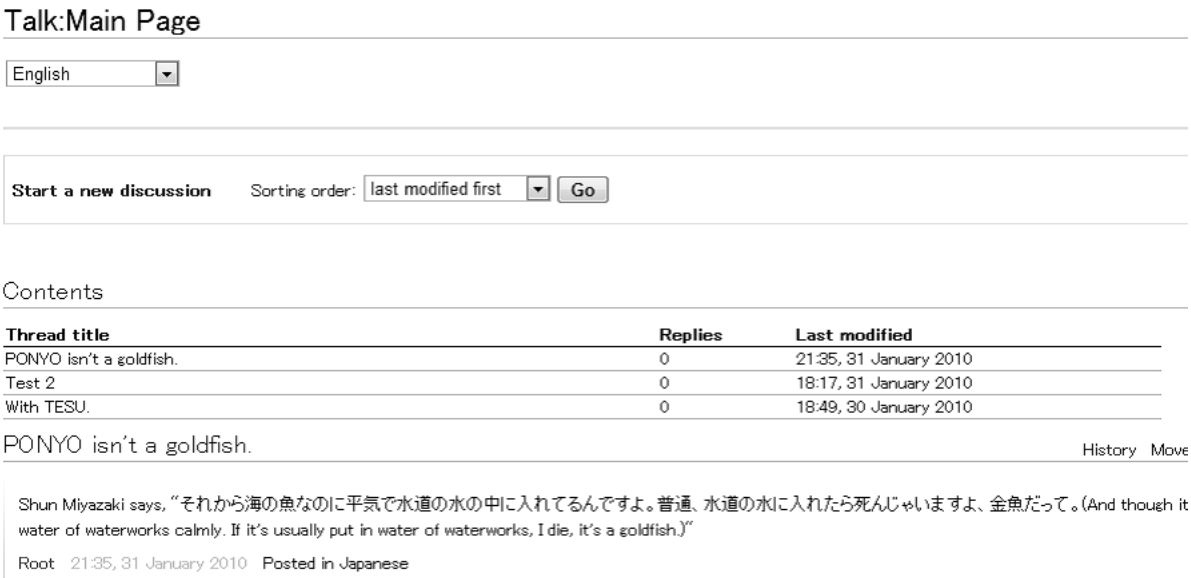

図 5.10: MediaWiki での動作例 (英語)

翻訳と，実装したサービスを適用した場合の翻訳で比較を行った．この際，第3章で述べた問題点ごとに分類し，改善されていることを実験によって確かめた．

6.3 改善例

6.3.1 翻訳されるべきでない部分が翻訳される問題

この場合においては，文の意図が変わってしまうことが問題となっていた．図6.1にサービスを適用した場合と適用しない場合の翻訳の比較結果を示す．

＜サービスを適用しない場合＞
“崖の上のポニョ”の英語での公式なタイトルは “Ponyo on a Cliff by the Sea.”です．
⇒ The formal title in English of “PONYO on the cliff” is “Ponyo on a Cliff by the Sea”.

＜サービスを適用した場合＞
“崖の上のポニョ”の英語での公式なタイトルは “Ponyo on a Cliff by the Sea.”です．
⇒ The formal title in English of “崖の上のポニョ(PONYO on the cliff)” is “Ponyo on a Cliff by the Sea.”

図 6.1: 文の意図が変わる問題の改善例

この例では，説明文として付加されている (PONYO on the cliff) は特に必要ではないが，サービスの適用によって日本語と英語の単語の関係が維持されていることがわかる．

また，入力者が翻訳されることを望まない場合については，サービスの適用によって，翻訳されない部分の意味の理解を助けることができる (図 6.2)．

この例でも，入力者が翻訳されることを望まない部分が日本語になっているが，説明文によって，日本語が理解できない人の理解を助けることができる．

さらに，原文が保存される性質を利用して，URL などが翻訳されてしまう場合にも対処できる (図 6.3)．

この例では，通常の翻訳では URL が利用できなくなってしまうが，サービス

＜サービスを適用しない場合＞
宮崎駿は，“それから海の魚なのに平気で水道の水の中に入れてるんですよ。普通、水道の水に入れたら死んじゃいますよ、金魚だって。”と言っています.
⇒Shun Miyazaki says, “And though it’s fish from sea, it’s kept in the water of waterworks calmly. If it’s usually put in water of waterworks, I die, it’s a goldfish.”

＜サービスを適用した場合＞
宮崎駿は，“それから海の魚なのに平気で水道の水の中に入れてるんですよ。普通、水道の水に入れたら死んじゃいますよ、金魚だって。”と言っています.
⇒Shun Miyazaki says, “それから海の魚なのに平気で水道の水の中に入れてるんですよ。普通、水道の水に入れたら死んじゃいますよ、金魚だって。(And though it’s fish from sea, it’s kept in the water of waterworks calmly. If it’s usually put in water of waterworks, I die, it’s a goldfish.)”

図 6.2: 入力者が翻訳を望まない場合の改善例

＜サービスを適用しない場合＞
Access “https://mail.google.com/mail/?shva=1♯inbox”.
⇒「https://mail.google.com/mail/？shva=1♯受信箱」にアクセスしなさい。

＜サービスを適用した場合＞
Access “https://mail.google.com/mail/?shva=1♯inbox”.
“https://mail.google.com/mail/?shva=1♯inbox(https://mail.google.com/mail/？shva=1♯受信箱)”にアクセスしなさい。

図 6.3: URL が翻訳されてしまう場合の改善例

を適用すると，原文が保存されることによってURLが利用可能になる．

6.3.2 複数回翻訳されてしまう問題

この場合は，一度機械翻訳された文が翻訳対象になることで，文の理解が困難になっていくことが問題となっていた．図6.4にサービスを適用した場合と適用しない場合の翻訳の比較結果を示す．

＜サービスを適用しない場合＞
宮崎: “実は、最初はカエルでいこうかと思ったんです。”
⇒ Miyazaki: “I thought actually, the beginning went by a frog.”
⇒宮崎： “私は、実際、最初がカエルによって過ぎたと思った。”

＜サービスを適用した場合＞
宮崎: “実は、最初はカエルでいこうかと思ったんです。”
⇒ Miyazaki: “実は、最初はカエルでいこうかと思ったんです。(I thought actually, the beginning went by a frog.)”
⇒宮崎: “実は、最初はカエルでいこうかと思ったんです。”

図6.4: 入力者が翻訳を望まない場合の改善例

この例では，日本語の文が英語に翻訳され，その後，その英文が再度日本語に翻訳されているが，サービスを適用すると，最初の日本語を原文として翻訳するようになるため，複数回翻訳される問題を解決することができる．

6.3.3 他の言語の語句が翻訳されない問題

この場合は，文の中に他の言語の語句が混じることで，その部分の言語が認識できないようになっていることが問題となっていた．図6.5にサービスを適用した場合と適用しない場合の翻訳の比較結果を示す．

この例では，文が日本語からドイツ語に翻訳されているが，“Door”は英語であるため，通常は翻訳することができない．そこで，サービスを適用し，例のように言語を指定すると，翻訳することができるようになる．

＜サービスを適用しない場合＞
“ドア”は英語で “Door”と書かれる．
⇒ “Tür” wird als “Door” in Englisch geschrieben.

＜サービスを適用した場合＞
“ドア”は英語で “<en>Door</en>”と書かれる．
⇒ “Tür” wird als “<en>Door</en>(Tür)” in Englisch geschrieben.

図 6.5: 他の言語の語句を含む場合の改善例

この言語指定の機能は，引用などの場面ではより重要である．これは，説明文を付加するのに機械翻訳を用いているためである．

＜言語指定タグを利用しない場合＞
Hayao Miyazaki says, “それから海の魚なのに平気で水道の水の中に入れてるんですよ。普通、水道の水に入れたら死んじゃいますよ、金魚だって。”
⇒ Hayao Miyazaki sagt, “.....”.

＜言語指定タグを利用した場合＞
Hayao Miyazaki says, “<ja> それから海の魚なのに平気で水道の水の中に入れてるんですよ。普通、水道の水に入れたら死んじゃいますよ、金魚だって。</ja>”
⇒ Hayao Miyazaki sagt, “<ja> それから海の魚なのに平気で水道の水の中に入れてるんですよ。普通、水道の水に入れたら死んじゃいますよ、金魚だって。</ja> (Dann obwohl es ein Fisch des Meeres ist, kann ich ins Laufen hineingehen, träne unbesorgt. Wenn es es gesetzt wird ins Laufen, trane im allgemeinen; sogar als für der Tod.)”

図 6.6: 他の言語の文を引用する場合の改善例

図 6.6 においては，言語指定タグを利用しない場合，引用部分が全く翻訳されず問題となっているが，言語指定タグを利用することで，説明文を付加することができる．

第7章　議論

前章では，実際にアルゴリズムを適用した場合に，様々な問題を解決できることを示した．しかし，翻訳のための議論において，アルゴリズムを適用しても問題を解決できないような場合もある (図 7.1).

“崖の上のポニョ”は英語で何に翻訳されるか.
⇒ What is “崖の上のポニョ(PONYO on the cliff)” translated into?

図 7.1: 問題が解決できない場合

この例では，第 3 章で挙げたような問題は発生していないが，この機械翻訳を見て適切な回答が得られる可能性は高いとは言えない．これは，説明文として与える情報が不十分であることに起因する．このような問題を解決するには，機械翻訳の精度を向上させるか，別の情報を与えることが考えられる．この例では，Wikipedia の記事から抽出できる辞書のエントリを用いて辞書連携翻訳すると，より適切な情報を得られる可能性がある.

第8章　おわりに

機械翻訳を用いて，翻訳のための議論を多言語で行うことを考えた場合に発生する問題として，翻訳されるべきでない部分が翻訳されてしまう問題が発生する．この問題は，上に述べたように，指定されない部分を翻訳されないようにする翻訳の回避によって解決できる．しかし，この操作によって別の問題が発生する．それは，翻訳結果に自国語ではない部分が含まれてしまい，理解ができなくなる問題である.

また，翻訳のための議論を行う場合に発生する別の問題として，何度も引用されることによって何度も機械翻訳され，文が理解困難なものになっていくというものや，他の言語の語句が混じる文において翻訳ができないというものがある.

本論文では，これらの問題を解決する方法として，4.5 節のアルゴリズムを示した．このアルゴリズムの基本的な動作は，翻訳を回避し，その部分に説明文を加えるというものである．これによって，翻訳されるべきでない部分が翻訳さ

れてしまう問題は解決できる．また，複数回翻訳される問題については，アルゴリズムの原文を保存する性質を利用して，その原文の翻訳を結果とすることで解決している．さらに，他の言語の語句が翻訳されない問題については，言語指定タグで囲むことによって解決できている．

さらに，Wikipedia 上で議論を行うための拡張機能として LiquidThreads というものがあるが，LiquidThreads を多言語化した Multilingual LiquidThreads に今回のアルゴリズムを組み込み，実際に動作している例を示した．

謝辞

本論文の作成に協力いただいた，石田・松原研究室の教員方，先輩および研究員の方々に深甚の謝意を表する．

参考文献

[1] Flournoy, S. R. and Callison-Burch, C.: Secondary Benefits of Feedback and User Interaction in Machine Translation Tools, Workshop paper for “ MT2010: Towards a Roadmap for MT ”of the MT, Summit VIII, pp. 2–3 (2001).

[2] Ishida, T.: Language Grid: An Infrastructure for Intercultural Collaboration, *IEEE/IPSJ Symposium on Applications and the Internet*, pp. 96–100 (2006).

[3] 吉岡真治: 多言語ニュースの対照分析のための Wikipedia 活用手法の研究, *The 23rd Annual Conference of the Japanese Society for Artificial Intelligence*, pp. 3–4 (2009).

[4] Eytan Adar, M. S. and Weld, D. S.: Information Arbitrage Across Multilingual Wikipedia, *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, p. 96 (2009).

付録：ソースコード

今回実装したプログラムのソースコードを以下に添付する．

A.1 MetaTranslator.php

MetaTranslator.php は，今回実装した翻訳の回避，および情報の付与に関する動作を実現するプログラムである．

```
<?php

require_once('SOAP/Client.php');

class MetaTranslator {

    private $sourceLanguageCode;
    private $targetLanguageCode;
    private $sourceText;
    private $soap_options;
    private $logOptions;
    private $wsdl_url;
    private $wsdl_func;
    private $wsdl_param;

    private $langridClient;

    private $TAG_BEGIN='[tag]';
    private $TAG_END='[/tag]';

    /**
     * Constructor
     *
     * @param String $sourceLanguageCode
     * @param String $targetLanguageCode
```

```
 */
public function __construct($sourceLanguageCode='', $targetLanguageCode='',
                                    $soap_options, $wsdl, $s_func, $s_param) {
    $this->setSourceLanguageCode($sourceLanguageCode);
    $this->setTargetLanguageCode($targetLanguageCode);
    $this->soap_options=$soap_options;
    $this->wsdl_url=$wsdl;
    $this->wsdl_func=$s_func;
    $this->wsdl_param=$s_param;
}


/**
 * translate
 */
public function translate($sourceText, $mode) {

    if (!$sourceText) {
        return;
    }

    $src=$this->getSourceLanguageCode();
    $tgt=$this->getTargetLanguageCode();

    $result=$this->makeTranslation($sourceText, $src, $tgt, $mode);


    return $result;
}

/**
 * make translation obj
```

```
 */
private function makeTranslation($sourceText, $src, $tgt, $mode) {
    $tag=null;

    $sourceText=stripslashes($sourceText);

    /*
     * separating tag
     */
    if($mode=='strict') {
        $tag=$this->tagSeparate_strict($sourceText);
    }
    if($mode=='relaxed') {
        $this->TAG_BEGIN='"';
        $this->TAG_END='"';
        $tag=$this->tagSeparate_relaxed($sourceText);
    }

    /**
     * remove discription
     */
    for($i=0; $i<count($tag); $i++) {
        if($tag[$i]['noTrans']!=null) {
            $tag[$i]['noTrans']=
                            $this->removeDiscription($tag[$i]['noTrans']);
        }
    }

    /*
     * set alternative string
     */
    for($i=0; $i<count($tag); $i++) {
```

```
            if($tag[$i]['Trans']==null) {
                $tag[$i]['Trans']=$this->setAlt($tag[$i]['noTrans']);
            }
        }

        /*
         * translation string
         */
        $trans=$this->joinTrans($tag);


        /*
         * do translate
         */
        $result=$this->doTranslate($trans, $src, $tgt);

        /*
         * set discription
         */
        $tag=$this->setDiscription($tag, $mode);

        /*
         * replace alternative
         */
        $text=$result;
        $alt=$this->replaceAlt($text, $tag);

        $result=stripslashes($alt);

        return $result;
    }
```

```
/**
 * replace alternatives
 */
private function replaceAlt($text, $tag) {
    $result=$text;

    for($i=0; $i<count($tag); $i++) {
        if($tag[$i]['noTrans']!=null) {
            $replace=$tag[$i]['noTrans'];

            //discription
            if($tag[$i]['Dis']!=null) {
                $replace.='('.$tag[$i]['Dis'].')';
            }

            $replace=$this->TAG_BEGIN.$replace.$this->TAG_END;

            $result=str_replace($tag[$i]['Trans'], $replace, $result);
        }
    }

    return $result;
}

/**
 * same except space & discription
 */
private function isSimilar($textA, $textB) {

    //remove discription
    $compA=$this->removeDiscription($textA);
    $compB=$this->removeDiscription($textB);
```

```
//remove punctuation
$buf='';
for($i=0; $i<mb_strlen($compA); $i++) {
    $chr=mb_substr($compA, $i, 1);

    if($chr!=' ' && $chr!='　' && $chr!='.' && $chr!='. ' &&
        $chr!='。' && $chr!=',' && $chr!=', ' &&
        $chr!='、' && $chr!='！' && $chr!='!' &&
        $chr!='?' && $chr!='？' && $chr!='：' && $chr!=':' &&
        $chr!=';' && $chr!='；') {

        $buf.=$chr;
    }
}
$compA=$buf;


$buf='';
for($i=0; $i<mb_strlen($compB); $i++) {
    $chr=mb_substr($compB, $i, 1);

    if($chr!=' ' && $chr!='　' && $chr!='.' && $chr!='. ' &&
        $chr!='。' && $chr!=',' && $chr!=', ' &&
        $chr!='、' && $chr!='!' && $chr!='！' && $chr!='？' &&
        $chr!='？' && $chr!='：' && $chr!=':' &&
        $chr!=';' && $chr!='；') {

        $buf.=$chr;
    }
}
$compB=$buf;
```

```
    return $compA==$compB;
}

/**
 * discription for noTrans
 */
private function setDiscription($tag, $mode) {
    for($i=0; $i<count($tag); $i++) {
        if($tag[$i]['noTrans']!=null) {
            $src=$tag[$i]['Lang'];

            /*
             * if lang-code is set
             */
            $code=$this->getLanguageCode($tag[$i]['noTrans']);
            if($code!=null) {
                $src=$code;
            }

            $tgt=$this->getTargetLanguageCode();
            $translation=$this->makeTranslation($tag[$i]['noTrans'],
                                                $src, $tgt, $mode);
            $transText=$translation;

            if(!$this->isSimilar($tag[$i]['noTrans'], $transText)) {
                $tag[$i]['Dis']=$transText;
            }
        }
    }
```

```
    return $tag;
}

/**
 * get language Code <ja></ja>
 */
private function getLanguageCode($text) {
    $tagBegin=mb_substr($text, 0, 4);
    $tagEnd=mb_substr($text, -5);

    /*
     * format check
     */
    $buf=mb_substr($tagBegin, 0, 1);
    if($buf!='<') {
        return null;
    }
    $buf=mb_substr($tagBegin, -1, 1);
    if($buf!='>') {
        return null;
    }
    $buf=mb_substr($tagEnd, 0, 2);
    if($buf!='</') {
        return null;
    }
    $buf=mb_substr($tagEnd, -1, 1);
    if($buf!='>') {
        return null;
    }

    /*
```

```
     * lang code
     */
    $buf=mb_substr($tagBegin, 1, 2);
    $buf2=mb_substr($tagEnd, 2, 2);
    if($buf!=$buf2) {
        return null;
    }else {
        return $buf;
    }
}


/**
 * real process
 */
private function doTranslate($sourceText) {
    $url=$this->wsdl_url;

    $option=$this->makeParam($this->soap_options);


    $client=new SOAP_Client($url, true, false, $option);


    $params=$this->makeParam($this->wsdl_param);
    if($params['sourceLang']=='$sourceLang') {
        $params['sourceLang']=$this->getSourceLanguageCode();
    }
    if($params['targetLang']=='$targetLang') {
        $params['targetLang']=$this->getTargetLanguageCode();
    }
    if($params['source']=='$source') {
```

```
        $params['source']=$sourceText;
    }


    $result=$client->call($this->wsdl_func, $params);

    return $result;
}


/**
 * parameter
 */
private function makeParam($param) {
    $result=null;

    $buf=split(';', $param);



    $names=array();


    for($i=0; $i<=count($buf); $i++) {
        if($i==count($buf)) {
            $result=compact($names);
            break;
        }


        $split=split(':', $buf[$i]);
        $names[$i]=trim($split[0]);
```

```
        ${trim($split[0])}=trim($split[1]);
    }

    return $result;
}

/**
 * array join
 */
private function joinTrans($array) {
    $result='';

    for($i=0; $i<count($array); $i++) {
        $result.=$array[$i]['Trans'];
    }

    return $result;
}

/**
 * set alternative text
 */
private function setAlt($text) {
    $alt=bin2hex($text);
    $result='';

    $alt2='';
    $length=mb_strlen($alt);
    $num=floor($length/18+1);
    for($i=0; $i<$length; $i++) {
        if($i%$num==0) {
```

```
            $alt2.=mb_substr($alt, $i, 1);
        }
    }

    $search=array('0','1','2','3','4','5','6','7','8','9');
    $replace=array('g','h','i','j','k','l','m','n','o','p');

    $result=str_replace($search, $replace, $alt2);

    return 'xxx'.$result.'xxx';
}

/**
 * remove discription : pre-trans
 */
private function removeDiscription($buf) {
    $disIndent=0;
    $str='';
    for($i=0; $i<mb_strlen($buf); $i++) {
        $chr=mb_substr($buf, $i, 1);

        if($chr=='(') {
            $disIndent++;
            continue;
        }

        if($disIndent==0) {
            $str.=$chr;
        }

        if($chr==')') {
            $disIndent--;
```

```
        }
    }

    return $str;
}

/**
 * separate tags : relaxed
 */
private function tagSeparate_relaxed($text) {
    $tag=array();

    $buf=split('"', $text);

    for($i=0; $i<count($buf); $i++) {
        $tag[$i]=array('noTrans'=>null, 'Trans'=>null, 'Dis'=>null,
                                    'Lang'=>$this->getSourceLanguageCode());

        if($i%2==0) {
            $tag[$i]['Trans']=$buf[$i];
        }else {
            $tag[$i]['noTrans']=$buf[$i];
        }
    }

    if($tag[0]['noTrans']==null && $tag[0]['Trans']==null) {
        $tag=array_slice($tag, 1);
    }

    if($tag[count($tag)-1]['noTrans']==null &&
                                     $tag[count($tag)-1]['Trans']==null) {
        $tag=array_slice($tag, 0, count($tag)-1);
```

```
    }



    return $tag;
}

/**
 * separate tags
 */
private function tagSeparate_strict($text) {
    $tag=array();
    $tagIndent=0;
    $id=0;
    $buf='';

    $i=0;
    while($i<mb_strlen($text)) {
        $chr=mb_substr($text, $i, 1);

        $comp=mb_substr($text, $i, mb_strlen($this->TAG_BEGIN));
        if($comp==$this->TAG_BEGIN) {
            if($tagIndent==0) {
                $tag[$id]=array('noTrans'=>null, 'Trans'=>null,
                      'Dis'=>null, 'Lang'=>$this->getSourceLanguageCode());

                if($buf!='') {
                    $tag[$id]['Trans']=$buf;
                    $buf='';
                    $id++;
                }
            }else {
```

```
            $buf.=$comp;
        }

        $i+=mb_strlen($this->TAG_BEGIN)-1;
        $tagIndent++;
    }else {
        $comp=mb_substr($text, $i, mb_strlen($this->TAG_END));
        if($comp==$this->TAG_END) {
            $tagIndent--;
            $i+=mb_strlen($this->TAG_END)-1;

            if($tagIndent==0) {
                $tag[$id]=array('noTrans'=>null, 'Trans'=>null,
                  'Dis'=>null, 'Lang'=>$this->getSourceLanguageCode());

                $tag[$id]['noTrans']=$buf;
                $buf='';
                $id++;
            }else {
                $buf.=$comp;
            }
        }else {

            $buf.=$chr;
        }
    }

    $i++;
}
if($buf!='') {
    $tag[$id]=array('noTrans'=>null, 'Trans'=>$buf);
}
```

```
        return $tag;
    }

    /**
     * getter/setter
     */
    public function getSourceLanguageCode() {
        return $this->sourceLanguageCode;
    }
    public function setSourceLanguageCode($sourceLanguageCode) {
        $this->sourceLanguageCode = $sourceLanguageCode;
    }
    public function getTargetLanguageCode() {
        return $this->targetLanguageCode;
    }
    public function setTargetLanguageCode($targetLanguageCode) {
        $this->targetLanguageCode = $targetLanguageCode;
    }
    public function getSourceText() {
        return $this->sourceText;
    }
    public function setSourceText($sourceText) {
        $this->sourceText = $sourceText;
    }
}
?>
```

A.2 MetaTranslatorServer.php

MetaTranslatorServer.php は，MetaTranslator.php で実現されるアルゴリズムを SOAP から Web サービスとして呼び出せるようにするためのプログラムである．

```
<?php
require_once( 'SOAP/Value.php' );
require_once( 'SOAP/Fault.php' );
require_once( 'SOAP/Server.php' );

function myCallHandler( $method, $args ) {
    global $soapclass;
    return @call_user_func_array( array( $soapclass, $method ), $args );
}



class MetaTrans {
    var $__dispatch_map=array();

    function MetaTrans() {
        $this->__dispatch_map['meta'] = array(
            'in' => array( 'sourceLang'=>'string',
                           'targetLang'=>'string',
                           'source'=>'string',
                           'mode'=>'string',
                           'soap_options'=>'string',
                           'service_wsdl'=>'string',
                           'service_function'=>'string',
                           'service_parameters'=>'string'),
            'out' => array( 'translateReturn' => 'string' ),
            'fault' => array( 'translateException' => 'string' )
            );
    }
```

```
    function __dispatch( $method ) {
        if ( isset( $this->__dispatch_map[ $method ] ) )
            return $this->__dispatch_map[ $method ];
        return NULL;
    }

    function meta($sourceLang, $targetLang, $source, $mode, $soap_options,
                      $service_wsdl, $service_function, $service_parameters) {
        try {

            require_once('./MetaTranslate.php');

            $mt=new MetaTranslator($sourceLang, $targetLang, $soap_options,
                        $service_wsdl, $service_function, $service_parameters);


            $result=$mt->translate($source, $mode);


            return new SOAP_Value( 'translateReturn', 'string', $result);
        }catch (Exception $e ) {
            return new SOAP_Fault( 'translate return: '.$e->getMessage() );
        }
    }
}


$server = new SOAP_Server();
$server->_auto_translation = true;
$soapclass = new MetaTrans();
```

```
$server->setCallHandler( 'myCallHandler', false );
$server->addObjectMap( $soapclass, 'MetaTrans' );

if( isset( $_SERVER['REQUEST_METHOD'] )
                                    && $_SERVER['REQUEST_METHOD']=='POST' ) {
    $server->service( $HTTP_RAW_POST_DATA );
} else {
    require_once( 'SOAP/Disco.php' );
    $disco = new SOAP_DISCO_Server( $server, 'MetaTranslation' );
    header( "Content-type: text/xml" );
    if( isset($_SERVER['QUERY_STRING'] )
                    && strcasecmp( $_SERVER['QUERY_STRING'], 'wsdl' ) == 0 ) {
        echo $disco->getWSDL();
    } else {
        echo $disco->getDISCO();
    }
    exit;
}
?>
```