E-040

Wikipedia翻訳のための多言語議論の支援

Supporting Multilingual Discussion for Wikipedia Translation

石田 憲幸† 高崎 俊之† 石松 昌展† 石田 亨†
Noriyuki Ishida Toshiyuki Takasaki Masanobu Ishimatsu Toru Ishida

1. 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．このとき，その 2 つのコミュニティに共通言語が無ければ，翻訳のための議論は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．

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 社が開発した 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，多言語の議論を行うためのシステムはいくつか開発されているが，翻訳のための議論を行うためのシステムはまだ開発されていない．

以下では，翻訳のための議論を多言語で行うことを考える．また，参加者が母国語で議論できるよう，機械翻訳を介して行う議論を考える．

2. 翻訳のための議論

まず，以下では，翻訳のための議論の参加者として，議論の対象に精通しているモノリンガルと，議論の対象に精通していないモノリンガルを仮定することにする．機械翻訳を介して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．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，機械翻訳によって文の意図が変わってしまう．

別の問題として，複数回翻訳される問題がある．議論という性質上，他人の発言を引用することがよくあるが，機械翻訳を用いた多言語議論の場合では，一度翻訳されたものがさらに翻訳される可能性がある．

さらに，多言語の議論の場面では，ある一連の発言の中に他の言語が含まれる可能性がある．機械翻訳では翻訳元言語と翻訳先言語を指定するが，複数の言語が含まれる場合，その言語対とは別の語句が翻訳されない問題が発生する．

以下では，この 3 つの問題について述べる．

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

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．その例として，図 1 のように，議論中において表記を尋ねる場合に発生する問題がある．(図中では，原文を別の言語へ翻訳する操作を⇒で表す．以下の図でも同様．)

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

図 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 2).

† 京都大学大学院情報学研究科社会情報学専攻

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

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'sfi sh 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 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 1 の場合は対応できるが，図 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．

2.2 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).

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　複数回の翻訳により理解が困難になる例

図 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．

2.3 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 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．
⇒ “Tur” wird als “Door” in Englisch geschrieben.

図 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．

3. 「翻訳のための議論」の翻訳

3.1 問題の解決手法

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 3 つの問題を解決するためのアルゴリズムを提案する．

1 つ目は，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が，これ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，翻訳の回避が行われることで，原文の言語が理解できない人には，その部分の意味が理解できないようになる．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．

2 つ目は，複数回翻訳されてしまう問題であるが，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．

3 つ目は，他の言語の語句が翻訳されない問題であるが，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．

3.2 アルゴリズム

3.1 節で述べた解決手法に基づくと，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．

これらの要件を元に，アルゴリズムの動作方針を以下のように設計した．

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. 翻訳の回避を行う際には，翻訳によって変化しない文字列(中間コード)で置き換える操作を行う．
7. 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．
8. 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．
9. 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．

実際に解決のためのアルゴリズムを記述すると，図 5 のようになる．

このアルゴリズムでは，まず翻訳の回避を行う部分をすべて中間コードで置き換え，機械翻訳してから中間コードを置き換え直すようになっている．置き返す文字列は，元の文字列にその説明文を付加したもので，説明文として元の文字列の翻訳を用いる．ここで，翻訳の回避を行う部分の中にさらに翻訳の回避を行う部分があった場合に対処するために，説明文を得るための翻訳には，このアルゴリズムの結果を用いる．

こ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，3.1 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 6 のように，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．つまり，引用文の翻訳を回避したうえで，引用文の中の一部をさらに回避する必要がある．

また，こ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，アルゴリズムの上で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．

Algorithm 1 *MetaTranslate*(α, X, Y)

α : 原文
X : 原文の言語
Y : 翻訳先の言語
Z : ある部分の言語
Replace(α, 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(β, X, Y):文字列 β を言語 X から言語 Y に翻訳する関数
getLang(α): 文字列 α に指定されている言語を取得する関数
[: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
Si (1≦i≦n+1) :翻訳の回避を行わない部分の文字列
Ti (1≦i≦n):翻訳の回避を行う部分の文字列
Ui (1≦i≦n): Ti に対応する中間コード(機械翻訳の影響を受けない)
Di (1≦i≦n): Ti に対応する説明文
+ : 文字列結合の演算子
←: 代入の演算子
Require: α = S1 +[+ T1 +]+ S2 +[+ T2 +]+... +[+ Tn +]+ Sn+1

```
for i =1 to n do
  α ← Replace(α, Ti, Ui)
end for
α ← Translate(α, X, Y)
for i =1 to n do
  Z ← getLang(Ti)
  Di ← MetaTranslate(Ti, Z, Y)
  α ← Replace(α, Ui, Ti+"("+Di+")")
end for
return α
```

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

図 5　動作アルゴリズム

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

図 6　入れ子が必要となる場合

4. MediaWiki 上での実装

4.1 MediaWiki

今回は MediaWiki の LiquidThreads を拡張する形でプログラムを実装した．まず MediaWiki の概要を述べる．

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 上で議論を行うための機能を提供している．LiquidThreads を利用すると，

Wikipedia の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．

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．

4.2 組み込みプログラム

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，翻訳してから元の文字列と説明文に置き換え直すようになっている．第 3 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．

ところで，今回のプログラムには，動作のモードがある．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緩和モード(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格モード(strict mode)である．

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．前者は，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 6 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，後者で入れ子に対応している．

他の機能として，言語を指定することもできる．これについては，<言語コード></言語コード>で囲むことで実現される．

4.3 Web サービス化

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．

実装した Web サービスの仕様としては，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．今回は，この Web サービスを MetaTranslationService として実装した．サービスのインターフェース仕様を次に示す(表 1)．

表 1　実装したサービスのインターフェース仕様

説明	引数に従って MetaTranslation を実行する．
パラメータ	sourceLang : 翻訳元言語コード targetLang : 翻訳先言語コード source : 入力文 mode : 動作モード (relaxed / strict) soap options : SOAP のオプション service wsdl : 翻訳サービスの WSDL service function : 翻訳メソッド名 service parameters : 翻訳メソッドの引数
返り値	翻訳結果

実装した サービスは，表 1 の仕様に基づいて SOAP から呼び出すことが可能である．

4.4 MediaWiki へのサービス組み込み

以下では，今回実装したサービスを MediaWiki に組み込むことについて考える．これ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．

今回のサービスは Multilingual LiquidThreads から呼び出される．そして，その中で翻訳サービスを呼び出し，アルゴリズムを実現する．現在の Multilingual LiquidThreads では翻訳の際に直接翻訳サービスを呼び出しているが，その代わりに今回のサービスを呼び出すようにすると，アルゴリズムが適用できる．

サービスは前処理，翻訳器呼び出し，後処理 の 3 つのフェーズに分けられており，前処理では中間コードへ置き換える作業，後処理では，中間コードを元に戻す作業が行われる．

4.5 辞書

今回実装したサービスの組み込みによって，翻訳のための議論を支援することができる．しかし，機械翻訳を介して議論を行うため，機械翻訳精度の問題を避けることはできない．機械翻訳精度を向上させるには，辞書を用いた翻訳(辞書連携翻訳)が有効である．今回は，Wikipedia 翻訳のための議論に用いるための辞書として，Wikipedia の記事を利用して辞書を抽出した．

MediaWiki 上では，拡張機能によって，言語グリッドを利用した辞書連携翻訳を行うことができる．Wikipedia の記事から辞書を抽出すれば，Wikipedia のコミュニティ内で有効利用できると考えられる．以下では，これの抽出方法について述べる．

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 7)．

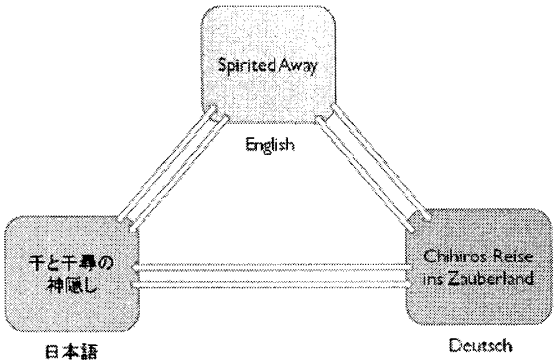

図 7　言語間リンク

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今回の辞書である．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:”の部分まで一致する必要があるが，そういった場合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．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不要なものを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. 評価

5.1 評価実験

今回実装したプログラムが実際に第 2 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．

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 2 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．

5.2 実験結果

まず，翻訳されるべきでない部分が翻訳される問題では，文の意図が変わってしまうことが問題となっていた．図 8 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 8　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．

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．URL の翻訳においては，通常の翻訳を用いると URL が利用できなくなってしまうが，サービスを適用すると，原文が保存されることによって URL が利用可能になる．

次に，複数回翻訳されてしまう問題というものがあったが，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 9 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 9　複数回翻訳される場合の改善例

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，複数回翻訳されることで文の理解が困難になる問題を解決することができる．

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 10 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＞
“ドア”は英語で“Door”と書かれる．
⇒ “Tur” 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．
⇒ “Tur” wird als “<en>Door</en>(Tur)” in Englisch geschrieben.

図 10　他の言語の語句を含む場合の改善例

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，例のように入力者が言語を指定すると，翻訳することができるようになる．

引用部分に対して説明文を付加する際に，その部分の言語が必要となるため，引用などの場面では特に重要な機能となる．

6. 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 11)．

“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 in English?

図 11　問題が解決できない場合

この例では，第 2 章で挙げたような，文の意図が変わってしまうなどの問題は発生していないが，この機械翻訳文だけ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．この文を見る人は一般には日本語を理解できないためである．この例の場合では，更なる議論の中で“崖の上のポニョ”に関する情報を得て回答を得ることが考えられる．説明文は，あくまで付加的な情報であり，これを回答のための直接の情報として利用することはできない．

7. 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，3.2 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，評価を行った．

謝辞

本研究は，SCOPE「戦略的情報通信研究開発推進制度」，グローバル COE プログラム「知識循環社会のための情報学教育研究拠点」(2007-2011) の補助を受けた．

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).