Bilingual KWIC — 対訳表現抽出の可視化による翻訳支援

小川 泰弘　外山 勝彦
名古屋大学情報基盤センター
yasuhiro@is.nagoya-u.ac.jp

1　はじめに

機械翻訳システムの性能向上や大量のコーパスを伴なう翻訳メモリなどの導入により，機械支援翻訳 (CAT) が広く行われるようになってきている．その一方で，翻訳の対象となる文書の内容が専門的である場合，その分野特有の専門用語や定型表現に関する対訳辞書が必要となる．そうした辞書を人手で作成することはコストが高いため，あらかじめ翻訳された対訳コーパスから専門用語や定型表現の対訳を自動抽出する研究が盛んである [1]．しかし，自動抽出の結果は必ずしも正確ではなく，間違った対訳表現を抽出したり，対訳表現の一部だけを抽出する場合がある．また，一つの語に対して複数の訳語を抽出した場合には，訳し分けに関する知見が必要となる．

そこで，対訳表現を抽出するだけでなく，対訳表現の各候補を，それが出現した文脈と一緒に表示することによって，ユーザによる対訳表現の選定を支援し，対訳辞書構築を支援するシステム Bilingual KWIC® を作成した．Bilingual KWIC は，対訳抽出の技術と KWIC(KeyWord In Context) 表示 [2] を統合し，文単位で対応付けされたパラレル・コーパスから，与えられたキーワードとその対訳表現の候補をそれぞれ文脈付きで表示する．そして，このシステムは対訳辞書構築だけなく，翻訳支援にも有効である．

本システムの目的は，従来の対訳辞書のようにあらかじめ登録された訳語と少数の用例を提示するのではなく，任意の入力キーワードに対して対訳表現を計算し，豊富なパラレル・コーパスからの情報を一緒に提示することにより，従来の対訳辞書や翻訳メモリとは異なるアプローチでの翻訳支援を実現することである．

2　Bilingual KWIC の概要

Bilingual KWIC の概観を図 1 に示す．これは Bilingual KWIC が「文脈検索」という名前で採用されている日本法令外国語訳データベースシステム (JLT)[1] [3] における画面である．

左上のキーワード入力欄にキーワードを入力し，その横の [検索] ボタンを押すと，左側に原言語，右側に対象言語で対応付けられた対訳文を表示する．その際，原言語ではキーワードを中心に，また，対象言語では自動的に推定したその対訳表現を中心に，それぞれ KWIC 形式で表示する．また，注目したい文をマウスでクリックすると，その文全体が下側に表示される．

右上の対訳表現入力欄には Bilingual KWIC が推定した対訳表現が表示されるが，それが間違っていたときには，ユーザが自分で適切な対訳表現をここに入力し，[訳語再検索] ボタンを押すことにより再表示が可能である．それに加えて，この欄右の [▼] ボタンを押すと，Bilingual KWIC が推定した他の対訳表現が表示され，ユーザは別の対訳表現を選択することが可能である．なお，キーワードおよび対訳表現ともにコーパス中における出現回数がすぐ横に表示され，対訳表現選定の一助となっている．

また，KWIC 表示欄の上部にある [並び替え] と表示された部分をクリックすることにより，出力結果をソートすることが可能である．原言語欄もしくは対象言語欄の中心にある表現の左側・右側でそれぞれソートすることができる．これにより対訳表現や用例の比較が簡単に行える．図 1 では，キーワード「専用利用権」の右側に続く語を基準にソートされている．

なお，図 1 では，原言語が日本語，対象言語が英語となっているが，キーワード入力欄に英単語を入力すれば，図 2 のように原言語を英語，対象言語を日本語として対訳表現の自動抽出が行える．またキーワード・対訳表現ともに複合語を含め，任意の文字列を入力することが可能である．

[1] http://www.japaneselawtranslation.go.jp/

図 1: Bilingual KWIC の概観

3 Bilingual KWIC の特徴

Bilingual KWIC は以下のような特徴を持つ.

1. 対訳表現の抽出における誤りの訂正が容易
2. 派生表現の獲得が容易
3. 訳し分けに関する知見の獲得が容易
4. 対訳辞書との併用が可能
5. 対訳表現を指定することが可能
6. 他の言語への応用が容易

Bilingual KWIC では，自動対訳抽出における誤りをユーザが簡単に修正できる．図 1 の例では，JLT で公開されている日本法令の日英対訳コーパス中を「専用利用権」をキーワードとして検索している．正しい対訳は，“exclusive exploitation right” であるが，自動対訳抽出の結果は “exclusive exploitation” となっている．しかし，図 1 の KWIC 形式で表示された英語文を見れば，“exclusive exploitation right” が正解であることが直観的に理解できる．

さらに，派生表現とその対訳を容易に獲得することができる．図 1 の例では,「専用利用権」“exclusive exploitation right” の下に続く用例から「専用利用権者」という日本語の派生語と，その対訳 “holder of an exclusive exploitation right” を得ることができる．

このような特徴は対訳表現の抽出を支援するときに有用であるが，対訳表現に加えて，実際の用例も表示されることから，Bilingual KWIC は翻訳支援に対しても有用である．特に訳し分けに関する知見が容易に獲得できる点が優れている．図 2 の例では，“negotiation” の対訳として,「譲渡」と「交渉」が表示されている．そのような場合には，前後の文脈から，対訳語がどのように使い分けられているかを比較することが容易であ

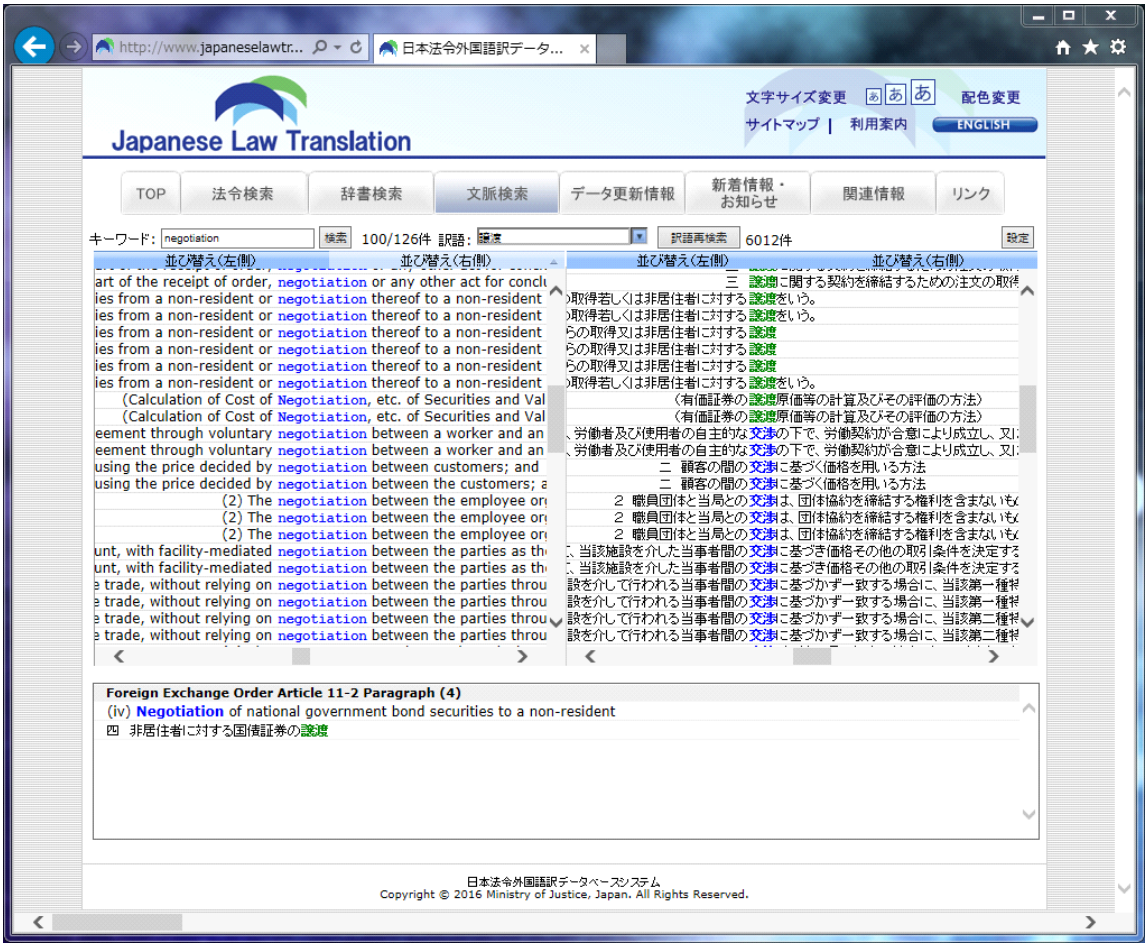

図 2: “negotiation” に対する実行結果

り，この例では “between” が後続する場合は「交渉」と翻訳するのが適当であるといった知識を得ることができる．

また，あらかじめ対訳辞書が用意されていれば，それを組み込んで併用することも可能である．辞書に対訳表現が登録されている場合は，その対訳を優先的に表示し，その後，自動的に推定した対訳表現を含む文を表示する．図 2 の例では，「譲渡」の文字が緑色になっているが，これは辞書に登録されている単語であることを示している．JLT においては，法令用語日英用標準対訳辞書が公開されており，JLT 上の Bilingual KWIC に組み込まれている．

対訳表現の抽出に失敗した場合や，特定の対訳表現に注目したい場合は，対訳表現入力欄にそれを入力することにより，対訳を指定できる．図 3 の例では，「検証」の訳語として少数ながら出現した “review” を入力することにより，その用例を表示している．なお，ユーザが指定した対訳表現は赤色の文字で表示され，辞書にある単語よりもさらに優先して表示される．

また，Bilingual KWIC は後述するように形態素解析を行わず，文字レベルの情報だけを利用しているため，様々な言語対での利用が可能である．図 4 は，ベトナム語と英語の対訳コーパスを利用した例である[2]．なお，この例のように，入力文字列から言語の判定が容易でない言語対で利用する場合は，入力言語を切り替えるトグルボタンをキーワード入力欄の左に用意している．

4 Bilingual KWIC の技術的詳細

本節では，Bilingual KWIC の技術的な詳細について述べる．まず，対訳表現の自動抽出手法について述

[2] これは JLT とは別のウェブサーバ上で実行した例である．

図 3: 「検証」の訳語として “review” を指定した結果

べたあと，どのような文字列を対訳ペアの単位とするかについて述べる．

4.1 対訳表現の自動抽出

4.1.1 Dice 係数の利用

対訳コーパスから対訳表現を自動的に抽出する手法については，これまでも様々な手法が提案されている．その中でも，現在広く利用されているものは 2 言語の単語間での対応付けを行う GIZA++[5] であろう．GIZA++は IBM モデル 1 からモデル 5 までを HMM に基づいて実装しており，出現回数がある一定以上の単語に関しては高い精度で対応を付けることが可能である．しかし GIZA++はあくまで単語間，正確にはあらかじめ分かち書きされた単位の間に対応を付ける．そのため単語の一部などのような表現に対しては対訳を推定することができない．また，あらかじめ対応関係を計算しておく必要がある．

GIZA++と異なり，まず対訳表現の候補を求め，候補間の類似度を計算する手法に関しては，文献 [1, 6] が詳しい．Bilingual KWIC では，文献 [6] において比較的高い精度をもつことが示された，以下の Dice 係数を類似度として採用した．

$$Dice(x, y) = \frac{2 \times freq(x, y)}{freq(x) + freq(y)} \quad (1)$$

ここで $freq(x)$ と $freq(y)$ は，入力キーワード x および対訳表現候補 y がそれぞれ原言語コーパスおよび対象言語コーパス中に出現する回数であり，$freq(x, y)$ は，対応付けられた文に x と y が同時に出現する回数である．よって，$0 \leq Dice(x, y) \leq 1$ となる．実際に Bilingual KWIC で使用する場合は，入力キーワード x を固定した上で，最大となる $\hat{y}$ を探すことになるこ

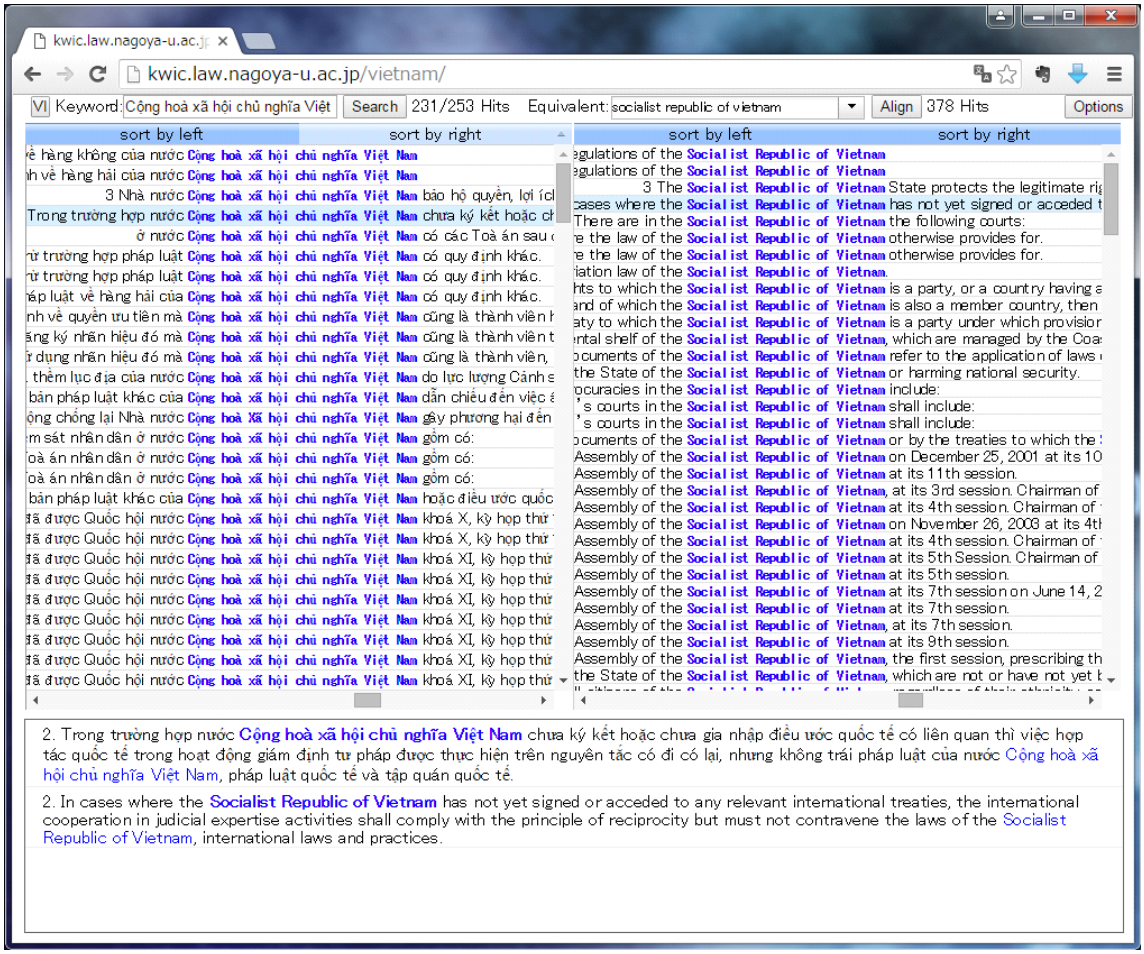

図 4: ベトナム語–英語コーパスへの適用

とから，以下の式を用いる．

$$\hat{y} = \arg\max_{y} \frac{2 \times freq(x, y)}{freq(x) + freq(y)} \tag{2}$$

実装においては，まずキーワード x が出現した文に対応する対訳文を集めて探索範囲とし，その中に出現するあらゆる候補 y について上記の式 (2) を計算し，最大となる $\hat{y}$ を求めている．

4.1.2 再帰的な対訳表現の抽出

上述の式 (2) を使用した場合，$\hat{y}$ は一つしか求められない．しかし，実際には同じキーワードが複数の対訳表現をもつことがある．そこで，Bilingual KWIC では，下記に示す方法で再帰的に複数の対訳表現を抽出する．

まず，最初の $\hat{y}$ が求まった場合，探索範囲から $\hat{y}$ を含んだ文を削除する．そして残った文を新たな探索範囲とし，再度式 (2) を計算することにより，異なる対訳表現を求める．探索範囲に含まれる文数や，$\hat{y}$ の Dice 係数の値が閾値以下になった場合は計算を終了し，そうでない場合はさらに再帰的に対訳表現を求める．

単純に Dice 係数の値が大きな順に対訳候補とすると，最初に求めた表現の部分文字列などが含まれる場合があるが，既に抽出した対訳表現が出現しない文から求めることにより，最初の候補とは別の対訳表現を抽出できる．

以上の方法により，図 2 のようにキーワードが複数の対訳表現をもつ場合に，それぞれを抽出することが可能である．

4.2 文字レベルの情報のみの利用

文献 [6] を含め，先行研究では日本語・英語とも形態素解析するものが多いが，Bilingual KWIC では，形態素解析をせずに文字レベルの情報だけを用いている．ここで文字レベルの情報とは，日本語の平仮名は対訳表現に含めない[3]，英語の単語は空白で区切られる，といった情報である．具体的には，日本語は文字 N グラム，英語は単語 N グラムを用いて，ある程度の長さ N をもつ対訳表現の候補を求めている．なお，N の最大値は言語ごとに指定可能である．

形態素解析を利用する利点として，対訳表現抽出の精度向上が期待できる点が挙げられる．特に動詞のように活用する語や，英語名詞の複数形などは形態素解析をしないと変化形が別の語として認識されてしまう．

その一方で，形態素解析での誤りが対訳抽出に影響を与える，他の言語に応用する場合はその言語に対応した形態素解析システムが必要といった問題もある．さらに，形態素解析システムの辞書にない単位では利用できないという問題がある．

そういった点を考慮し，さらに対訳表現抽出の誤りを容易に修正できることから，Bilingual KWIC では形態素解析を利用しないこととした．それにより，語の一部だけをキーワードとして入力するなど，柔軟な入力を可能にしている．ただし，単語の途中からを候補対象とすると，精度や速度の点で問題があるため，接頭語として含まれている場合だけを数え上げている．例えば，“search” の出現回数を数えるときには “searches”, “searching”, “searched”, “searcher” なども含めて数えている．これにより，動詞の活用形や名詞の複数形が規則変化する語については，形態素解析なしでも，ある程度対処できている．

5 Bilingual KWIC の開発

本節では，Bilingual KWIC の開発について，その段階にそって紹介する．

5.1 プロトタイプ版の開発

プロトタイプ版となる最初の Bilingual KWIC の実装は，筆者が独自に行った．PC 上でスタンドアロンで動作するように設計し，使用した言語は Ruby，GUI 作成のために Tk のライブラリを利用した．Ruby はスクリプト言語であり，プログラム開発が容易である一方，実行速度が遅いという欠点がある．Bilingual KWIC の実装においては，Dice 係数の計算のために文字列の出現回数を高速に数え上げる必要がある．そこで，文字列検索の高速化のために，Suffix Array[7] のライブラリである sary[4] を使用した．なお，今日では当然であるが，文字コードには UTF-8 を採用した．これにより，図 4 の例のように，各種言語に対応した．

5.2 対訳辞書作成支援版の開発

プロトタイプ版 Bilingual KWIC は，JLT[3] で公開される法令用語日英用標準対訳辞書の構築において利用された [4]．その際には，標準対訳辞書に収録する対訳ペアの候補を収集することが必要になるため，Bilingual KWIC 上から対訳ペアを登録可能とした．具体的には，登録したい単語を指定して右クリックすると図 5 のポップ・ウィンドウが表示されるようにし，必要に応じてデータを修正・追加して，登録できるようにした．

実際に，このシステムがインストールされたノート PC を作業担当者に配布することにより，標準対訳辞書に収録する対訳ペアの候補が収集された．この時点では，Bilingual KWIC が利用する対訳コーパスの大きさは 39,560 文であったが，ノート PC 上で問題なく動作していた．

ただし，この対訳ペア登録機能は，対訳辞書の構築には有用であるが，翻訳支援の目的とは異なるため，次節のウェブ対応版では採用しなかった．

5.3 ウェブ対応版の開発

プロトタイプ版はスタンドアロンな PC 上で起動するが，あらかじめ Ruby などを用意する必要があり，インストールが簡単ではなかった．また，利用者が各自で対訳コーパスを用意する必要があり，広く使用してもらうことができなかった．

そこで多くのユーザに利用してもらうために，ウェブサーバ上で動作可能なバージョンを開発した．基本的なエンジンはプロトタイプ版と同じであるが，ウェブブラウザを利用するための GUI 部分の作成は業者に委託した．

[3] オプションで含めることも可能．

[4] http://sary.sourceforge.net/

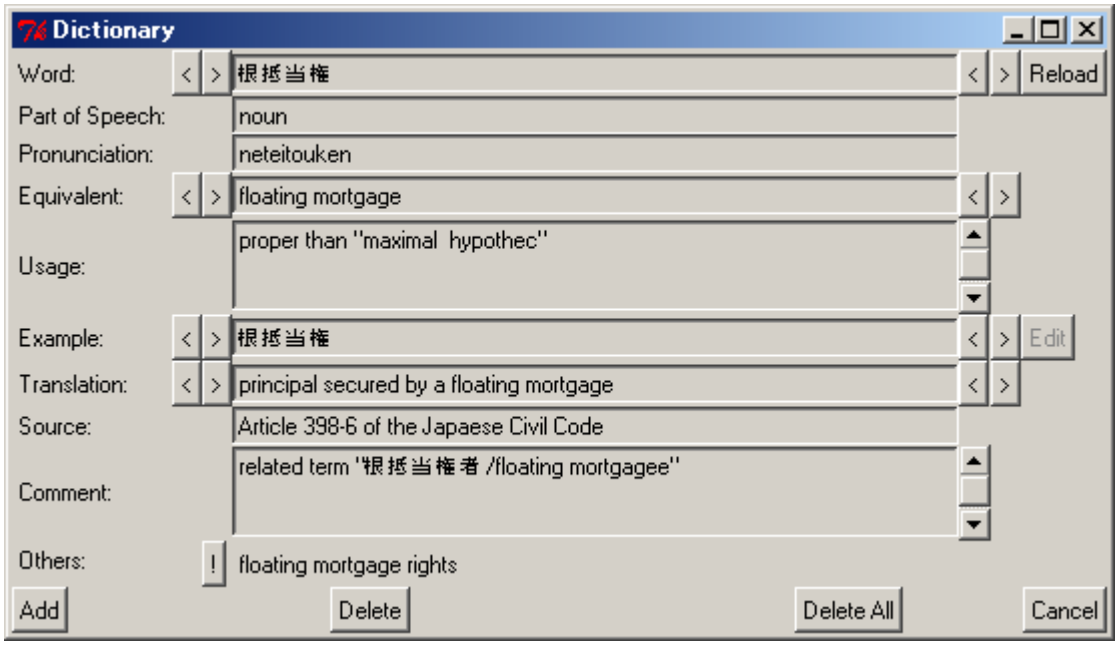

図 5: 対訳ペア登録用ウィンドウ

このウェブ対応版はさらに改良され，2006 年開設の日本法令外国語訳データベースシステム (JLT) において「文脈検索」の名前で採用されている．これにより多くのユーザに利用してもらうことが可能となった．

5.4 高速化

最初のプロトタイプ版では，対訳コーパスとして数万文程度の単語を想定していた．しかし，JLT では新しい法令の翻訳が次々に追加されるため，コーパスのサイズが 10 万文を超えるくらいの段階において，実行速度の点で問題が発生した．特に入力キーワードの出現回数が 1,000 を超えるような場合は，結果が表示されるまでに数十秒かかることもあった．

この問題に，以下の二つの方法で対処した．

5.4.1 対訳表現抽出の高速化

実行速度が遅い原因の一つは，コーパスが大きくなると対訳表現抽出に時間が掛かるためである．式 (2) においては，$\hat{y}$ を求めるために，あらゆる対訳表現候補 y について $freq(y)$ と $freq(x, y)$ の計算が必要となる．y は，上限 (デフォルトは日本語で 6 文字，英語で 4 単語) までのあらゆる N グラムを候補とするため，入力キーワード x が出現する文数が大きくなると，それだけ多くなる．このうち，$freq(y)$ の計算は，事前にコーパスの並び替えとインデックス化を行う Suffix Array により，高速に実行できるため問題ない．しかし $freq(x, y)$ に関しては，y の出現回数を数える探索範囲が x に依存して変化するため，事前にインデックス化が必要となる Suffix Array を用いた高速化ができない．そのためプロトタイプ版では，探索範囲内を順次 N グラム分割し，その出現回数を数え上げていた．

この $freq(x, y)$ の計算を高速化するため，コーパスの各文をあらかじめ一定サイズ以下の N グラムに分割したデータを別に保持し，それを数え上げることとした．デフォルトでは日本語は 8 文字以下，英語は 10 単語以下の N グラム単位であらかじめ分割したデータを保持しておく．また実装においては，テキストデータ自体へのアクセスを高速化するために Tokyo Cabinet[5] を導入した．

5.4.2 表示の高速化

実際の利用の上では，表示速度においてもボトルネックがあった．入力キーワード x が出現する文が多くなると，ブラウザ側に負担が掛かり，表示が遅くなって

[5] http://fallabs.com/tokyocabinet/

いた．そのため，デフォルトでは 100 文を超える分は表示しないようにし，オプションで上限を 200，400，800 と変更できるようにした．このため，キーワードを含む文のすべてを表示することはできない場合があるが，通常の利用では，800 文を表示できれば充分に比較ができると考えた．また，入力キーワードに文字列を付加したり，対訳表現を指定することにより，希望する対訳文を表示することが可能である．

また，最初の実装では，表示結果のソートをブラウザ側で実現していたが，これもブラウザに負担を掛け，表示が遅くなる結果となっていた．よってソート自体をサーバで実行し，その結果をブラウザ側に再送して再表示することにより，高速化を実現した．

6 まとめ

本稿では，対訳表現抽出を可視化することで翻訳を支援する Bilingual KWIC の開発について述べた．Bilingual KWIC は，任意の入力キーワードに対して対訳表現を自動抽出し，パラレル・コーパス中での用例と一緒に提示することにより，ユーザの翻訳を支援する．

本システムは，既に述べた通り，日本法令外国語訳データベースシステム (JLT) で採用されている他，名古屋大学が開発した学内情報翻訳データベース NUTRIAD[6][8] でも採用されている．NUTRIAD のシステムは，今後，九州大学などにも導入される予定であり，Bilingual KWIC も同様に利用される予定である．

現在，JLT では 36 万文 150MB 以上からなる対訳コーパスを用いて Bilingual KWIC を運用しているが，通常の使用では問題なく動作している．しかし出現回数が 1 万回を超えるようなキーワードに対しては，結果が表示されるまでに数十秒かかることがあり，Bilingual KWIC の一層の高速化が求められている．これに対しては，出現回数や入力頻度の多いキーワードに対する計算結果をキャッシュしておくなどの対応を検討している．

先行研究として，文献 [9] が挙げられる．ここに挙げられた翻訳メモリの利用では，与えられたキーワードの日本語コーパスにおける出現を KWIC 形式で表示し，コーパス中においてキーワードと良く共起した日本語単語と英語単語を提示する．ユーザが対訳候補となる英語単語を選ぶと，それに基づいた絞り込み検索を行い，対訳文のペアを別ウィンドウに表示する．ただし，別ウィンドウに表示される対訳文のペアでは，キーワードと対訳候補の部分がそれぞれ下線で明示されるが，中心に揃えて表示される訳ではない．Bilingual KWIC では，最初に対訳候補を自動的に決定する，任意の対訳表現を指定できる，原言語だけでなく対象言語も同時に KWIC 形式で表示する点が異なる．

謝辞 Bilingual KWIC の開発にあたっては，ウェブ版インターフェイスの開発，高速化などにおいて株式会社リーガルアストレイに協力をいただいた．

[6] http://nutriad.provost.nagoya-u.ac.jp/

参考文献

[1] Mastumoto,Y. and Utsuro, T.: Lexical Knowledge Acquisition, *Handbook of Natural Language Processing*, Dale, R., Moisl, H., and Somers, H.(Eds.), pp. 563–610, Marcel Dekker (2000).

[2] H.P. Luhn, “Key-Word-In-Context Index for Technical Literature (KWIC Index),” *American Documentation*, Vol. 11, No. 4, pp. 288–295 (1969).

[3] 外山勝彦, 齋藤大地, 関根康弘, 小川泰弘, 角田篤泰, 木村垂穂, 松浦好治: 日本法令外国語訳データベースシステムの設計と開発, 情報ネットワーク・ローレビュー, Vol. 11, pp. 33–53 (2012).

[4] 外山勝彦, 小川泰弘: 自然言語処理の応用に基づく法令外国語訳支援, 人工知能学会誌, Vol. 23, No. 4, pp. 521–528 (2008).

[5] Och, F.J. and Ney, H.: A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, Vol. 29, No. 1, pp. 19–51 (2003).

[6] 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).

[7] 山下 達雄: 用語解説 「Suffix Array」, 人工知能学会誌 Vol. 15, No. 6, p. 1142 (2000).

[8] 福田 薫, 外山 勝彦, 野田昭彦：学内情報翻訳データベースの構築と運用, 大学 ICT 推進協議会 2013 年度年次大会論文集, pp. 146–152 (2013).

[9] 内山 将夫, 井佐原 均: 日英新聞記事対応付けデータを用いた翻訳メモリと言語横断検索, 情報処理学会第 65 回全国大会第 5 分冊, pp. 355–358 (2003).