大規模テキストコーパスを対象にした対訳テキスト抽出の高速化

斎藤 大 † 田浦 健次朗 ‡ 近山 隆 †

† 東京大学新領域創成科学研究科 ‡ 東京大学情報理工学系研究科

{std, tau, chikayama}@logos.ic.i.u-tokyo.ac.jp

1 背景と目的

対訳テキストは，統計的機械翻訳や対訳辞書・シソーラスの構築など多言語自然言語処理の分野において非常に有用な資源である．ここで対訳テキストとは複数の言語で記述され意味内容が同じテキストのペア，いわゆる翻訳の形となっているものを指す．対訳テキストを収集し利用可能な形式にしたものは対訳コーパスと呼ばれるが，一般に十分な量・種類の対訳コーパスを入手することは困難であり，また人手で新たなコーパスを構築するのはコストが非常にかかってしまう．

そこで近年 Web から対訳テキストを発見・収集しコーパスを構築する手法が提案されている．Web 上には英語を中心に世界中の様々な言語で記述されたページが公開されており，テキスト中で扱われているジャンルも多岐に渡っている．有名な物語の翻訳やニュース記事の翻訳等多言語で記述・公開されているものが多数存在すると考えられ，Web から対訳ページを自動的に収集することで巨大なコーパスが構築出来ると期待される．しかしリソースとして非常に巨大である反面，対訳ではないページも多数存在するため，単純に Web 上の全ページのあらゆるペアに対して対訳判定を行うのは現実的ではない．またこれまでの対訳テキスト抽出手法のように，URL や HTML 構造などの表層的情報のみに頼ると多くの潜在的な対訳テキストが見逃されてしまう可能性がある．そこで本稿では，Web 上の大量のテキストから構成されたテキストコーパスを対象に対訳候補を高速に見つける手法として，サンプリングによるテキスト振り分け・LDA による次元削減・転置 index による高速化を組み合わせた手法を提案する．

2 関連研究

2.1 対訳テキスト抽出

テキストコーパスから対訳テキストを抽出する場合，大きく分けて

- 対訳テキスト候補を発見
- テキストペアの対訳判定

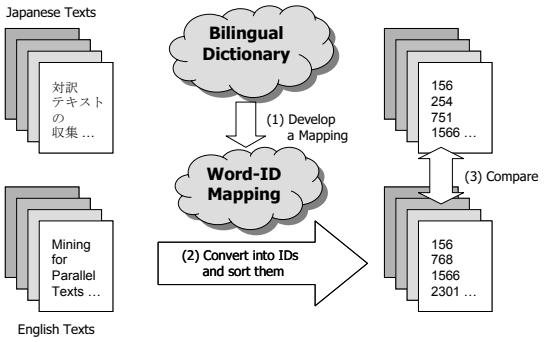

図 1 Flow of detecting parallel texts

の二つの段階が考えられる．理想的には Web 上のテキスト全てを対象として対訳テキストの発見を行いたいが，Web 空間の大きさを考えると全てのページを探索することは単純な比較だけでは事実上不可能である．そこで対訳テキストが高確率で存在しそうな範囲に絞って計算量を削減している．既存手法では，テキストを発見する段階で URL や検索クエリ等で絞込みを行っていたり [3, 6]，特定のホスト内の一部ページをダウンロードして対訳が存在すると判定されたらそのホストから全てをダウンロードするといった手法 [5, 7] が用いられている．またテキストを対訳と判定する段階では，候補ペアに対して厳密な判定を行う．対訳辞書を用いたテキスト類似性判定 [5] や URL 文字列・HTML ツリーの類似性を用いた判定 [6]，HTML 構造を用いた判定 [7] などが用いられている．初期段階で探索範囲を狭く絞れば次段階の対訳判定に精密な判定を施せるが，全体として手に入るテキスト数は少なくなる．また探索範囲を広げすぎると膨大な計算量となってしまう．

2.2 対訳判定

福島らの手法 [4] は，図 1 のようにテキストを数列に変換し比較を行うことで高速な対訳判定を実現している．ただし実際の対訳テキスト収集については全対全比較以上の言及はなされていない．

2.2.1 テキストの数列表現

テキストを高速に比較するための数列表現について説明する．まず事前に対訳辞書に対して，単語をノード・翻訳関係をエッジとしたグラフ構造を構築する．このグラフ構

造中の接続された各グループに対して一意な ID を付ける．次にテキストから名詞を抽出してグラフ構造を元に ID へ変換する．存在しないノード，つまり辞書中に存在しない単語は無視する．EDR 電子化辞書[*1]を用いてグラフを構築したところ 9,448 の ID が生成された．これにより各テキストが約 9,000 次元の数列へと変換される．

2.2.2 テキスト間の類似度の定義

2 つのテキスト間の対訳度合いを示した指標を *tscore*(translational score) として定義する．対訳関係にある 2 つのテキスト中には同じ ID の単語が共通して出現するので，次のように tscore を定義することで対訳判定の指標とすることが出来る．

$$tscore = \frac{\text{2 つの数列間に共通する要素の数}}{\text{2 つの数列の要素数の和}}$$

ここで数列とはテキスト中の名詞 ID の数列である．$tscore > 1.02$ を満たすテキストを対訳関係とみなすことで最も良い精度が出た，と報告されている．ただし，[4] では tscore と定義しているがこの名前は 2 単語の共起関係の指標である *tscore*[2] と重なってしまうので，本稿では今後この指標を trans_score と呼ぶ．

3 提案手法

大規模テキストコーパスを対象にした対訳テキスト抽出の高速化の手法として，サンプリングによるテキスト振り分け・LDA による次元削減・転置 index による高速化を提案する．

3.1 サンプリングによるテキスト振り分け

テキスト集合の中からラベルとなるテキストをサンプリングし，ラベルとの距離を用いたクラスタリングを行う．これは，‘対訳となっているテキストペアは特定のテキストとの距離が同じくらい近い’ という仮定に基づいている．対訳関係にあるテキスト間の距離は他のテキストとの距離と比べて近いと考えられるためこのような仮定を置いた．各テキストとサンプルテキストとの距離を測定し近いラベルに振り分けることで，明らかに距離が遠いテキスト間の無駄な対訳判定が削減出来，計算コスト削減に繋がると期待される．

計算コストを削減する流れについて説明する．まず，テキスト数 n と比べて十分小さい数のラベルとなるサンプルテキストをランダムに用意する．距離の尺度として trans_score を用いるので，trans_score の値が高いほど距離が近いペアであるとみなすことになる．各テキストに対し

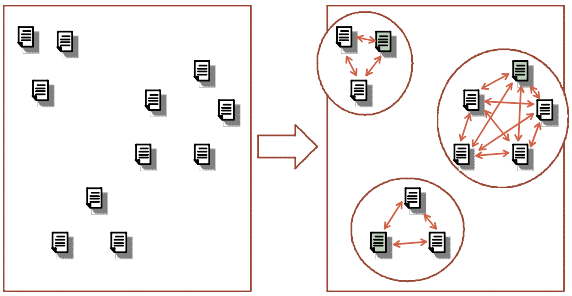

図 2 Sampling

てまずサンプルテキストとの距離を測定しその中で距離が近いもののラベルを付ける．理想的には最も距離が近いラベルに対して振り分ければ良いが，本来対訳であるものが誤ったラベルに振り分けられる危険を避けるための対処として，距離が近いラベルの上位いくつかに対して振り分けることも考えられる．この上位いくつかに振り分けるのかの値を，‘多重度’ として定義する．多重度 1 の場合の振り分けの概要を図 2 に示す．これにより全対全比較を行う場合に比べ対訳テキスト判定の回数が減ることから計算量の削減が期待される．

3.2 LDA による次元削減

対訳判定に重要な単語とトピック分類に重要な単語はほぼ同じであるとの仮定の下で，トピック分類に用いる文書生成モデルである LDA(Latent Dirichlet Allocation)[1] を用いて単語抽出による次元削減を行う．具体的には LDA によりトピック毎に単語の生成確率が求められるので，各トピックに対してその数値が高いものを重要な単語と位置づけ単語抽出を行う．

3.2.1 単語抽出

LDA による単語抽出の流れについて説明する．LDA は事前パラメータとしてトピック数を与えることで，データ集合に対して多重トピックに対応したモデル化を行い

1. 各トピックにおける単語 (意味 ID) の生成確率
2. 各トピックの混合割合

が出力される．特に今回扱うのは上の ‘各トピックにおける単語の生成確率’ であり，つまりトピック毎にそのトピックを特徴付ける語が抽出できることになる．提案手法ではこの ‘各トピックを特徴づける語’ が重要な単語であると定義し，トピック毎にこの生成確率が高い単語を抽出する．

3.3 転置 index を用いた対訳テキスト抽出の高速化

3.3.1 対訳テキスト検索

転置 index を用いた対訳テキスト検索の流れは以下の通りである．まずはクエリを意味 ID に変換する．ここでクエリとは対訳を探すために提示される元テキストのことを指

[*1] http://www2.nict.go.jp/kk/e416/EDR/J_index.html

す．クエリから形態素解析を用いて名詞を抽出し，対訳辞書から構築されたグラフを用いて意味 ID 変換を行う．次に意味 ID 列から転置 index 情報を引いてくることで，‘クエリテキストに出現する意味 ID がどのテキストにどれだけ出現するか’ のリストを生成することが出来る．転置 index 情報のリストをテキスト毎にまとめなおし，クエリテキストの対訳候補となるテキストをランク付けして出力する．各テキストのスコアを表す式は以下になる．

$$\sum_t (min(len(T_1,t), len(T_2,t)))/(len(T_1)+len(T_2))$$

ここで T_1, T_2 はそれぞれ各テキストの意味 ID 列であり，$len(T1)$ で T_1 の単語数，$len(T_1,t)$ で単語 t がテキスト T_1 に出現する数，$\sum_t$ は出現する単語全てについてという意味である．つまり，両方のテキストに含まれる各単語に対して，単語出現数の小さいほうの総和を各テキストの長さの和で割ったものとなる．この値を 2 テキスト間の対訳らしさの指標とし，今後 inv_score と呼ぶ．このようにすることで，trans_score から単語間距離の情報は落とされつつも，既にある程度性能が高いと示されている trans_score と近い指標を用いて計算することができる．

3.3.2 対訳テキスト抽出の高速化

次に，inv_score を用いて対訳テキスト抽出を行う．転置 index を用いた検索で ‘1 テキストに対してそれと似ているテキストのランク付け’ を高速に行うことが出来る．これをサンプリングに適用することで振り分けの高速化を図る．サンプリングの流れとして，大きく分けて各テキストをグループに振り分ける段階とグループ内で全対全比較を行い対訳テキストを抽出する段階があるが，後者は対訳テキストの抽出精度に厳密に関わってくるので trans_score を使い，前者の振り分けではそこまで厳密な指標を用いる必要がないため inv_score を用いて高速化する．具体的には，サンプリングでラベルとなるテキストを振り分ける段階でラベルをクエリをした対訳テキスト検索を行い，各テキストとラベルとの距離を測定する．それをテキスト毎にまとめなおしラベルでソートする．これにより距離の定義が ‘trans_score’ から ‘inv_score’ にはなるが ‘テキスト中の単語でクエリにも同時に存在する単語のみ’ に注目すれば良くなり全テキストとサンプルテキストとの距離を計算することが出来る．

4 実験

評価用のデータとして Fry[3] の WebCorpus の日英対訳データを用いた．また実験には CPU Xeon 2.4GHz, メモリ 2GB の Linux マシンを用いた．

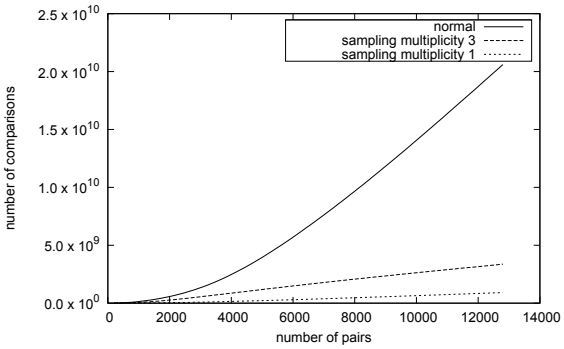

図 3 Execution time

4.1 サンプリングによる計算コスト削減

まずはサンプリングにより計算コストがどの程度削減されるかを示すために，Fry の WebCorpus の 12,800 ペアに対して対訳判定を行った．振り分ける条件は，サンプリングを行わない，多重度が 1，多重度が 3 の 3 つである．対訳判定はペア数を 200 から 12,800 まで 2 倍に増やしたそれぞれについて全対全比較を行った場合とサンプリングを行った場合で単語比較数を測定した．ここで単語比較数とは trans_score を計算する時に必要となる意味 ID の比較回数であり，trans_score 計算では最悪でも二つのテキストの意味 ID 数以内で比較回数が収まる．この数値は振り分けにかかる計算と振り分け後の全対全の trans_score 比較でかかる計算の合計であり，つまり計算コストと同義である．サンプル数は $\sqrt{n}$ とした．

振り分けによる計算コスト削減の計算結果を図 3 に示す．単純に全対全比較を行った場合の結果はグラフの上の線で，サンプリングによって振り分けを行い比較回数を減らした場合の結果はグラフの下の線で表される．実行時間が全対全で比較した場合と比べ小さくなっているのが分かる．12,800 ペアの場合は，多重度 3 の場合でおよそ 6 倍程度，多重度 1 の場合はおよそ 20 倍程度早くなっていることが分かる．テキスト数が増えるほど実行時間の差も大きくなっていることから，全対全比較がオーダ n^2 なのに比べ，オーダを小さく出来ていると思われる．

4.2 LDA による次元削減の評価

Fry のコーパス 6400 ペアに対して LDA をかけ，その出力からトピック分類に有用な意味 ID を 100, 200, 300, 400 語集める．それぞれの単語に対してサンプリングを行った時の計算量の変化を図 4 に，誤分類率の変化を図 5 に示す．ここで横軸はサンプル数である．これら二つのグラフが示すように，誤分類率は次元削減を行わない場合と行った場合でほとんど有意な差が見られないことが分かる．また，単語比較回数は扱う次元数が少なくなるほど小さくなっているのが分かる．LDA200 だとおよそ 1/2, LDA100 の場

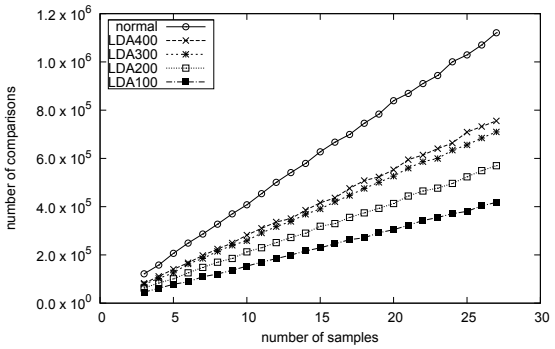

図 4　# of word comparison with LDA

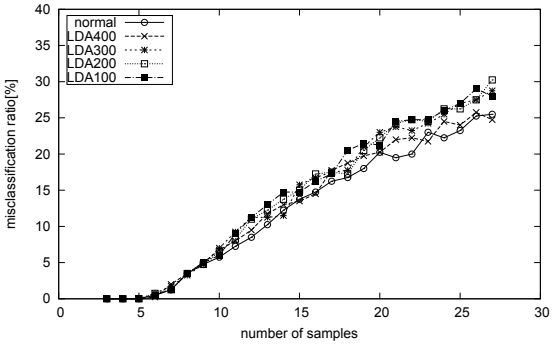

図 5　Misclassification ratio with LDA

合はおよそ 1/3 程度に減少している．この結果から，LDA を用いた次元削減を行うことで全体的な精度をほとんど落とさずに計算時間が削減され，振り分け時間の高速化が行われたと言うことが出来る．

4.3　対訳テキスト抽出の高速化の評価

サンプリングと転置 index，更に LDA を用いた単語空間 (単語数 400) での計算コストと誤分類率の関係を図 6 に示す．多重度を 1〜5 の範囲で定めそれぞれに対してサンプル数を 1〜28 の範囲で変えて行った時の単語比較数と誤分類率の関係をプロットした．また中央付近の縦線は全対全比較の計算コストである．通常のサンプリングに比べて転置 index を用いた方が，また転置 index に比べて LDA も用いた方がグラフが左下側に寄っているのが見てとれる．このグラフが左下によることは同等の時間でより精度が高いものが得られ，計算速度を向上させることで相対的に同等の時間での精度が良くなっている，と言うことが出来る．

5　まとめ

本稿では，大規模なテキストコーパスから対訳テキストを抽出するための手法としてサンプリングを用いた高速化・LDA による次元削減・転置 index による高速化について述べた．テキスト集合を振り分けることで大規模なものへの

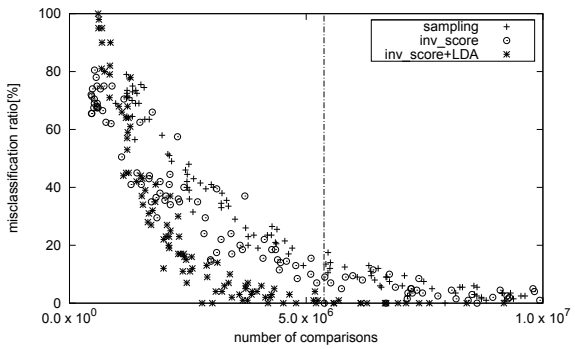

図 6　Number of comparisons v.s. Misclassification ratio

対応を可能とし，LDA を用いた次元削減と inv_score を用いることで同等の精度を示す場合の計算コストを削減した．今後の課題として，サンプリングのアルゴリズムをより精度の高いものに変えること・実際に様々な Web テキストに対して本手法を適用すること，等が挙げられる．

参考文献

[1] D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent Dirichlet Allocation, 2003.

[2] K.W. Church, W. Gale, P. Hanks, and D. Hindle. Using statistics in lexical analysis. In *Using On-line Resources to Build a Lexicon*, pp. 115–164, Lawrence Erlbaum, 1991.

[3] J. Fry. Assembling a parallel corpus from RSS news feeds. In *Proceedings of the Workshop on Example-Based Machine Translation*, MT Summit X, Phuket, Thailand, September 2005.

[4] K. Fukushima, K. Taura, and T. Chikayama. A fast and accurate method for detecting English-Japanese parallel texts. In *Proceedings of the COLING/ACL Workshop on Multilingal Language Resources and Interoperability (MLRI 2006)*, Sydney, Australia, July 2006.

[5] X. Ma and M. Liberman. BITS: A method for bilingual text search over the web. In *Machine Translation Summit VII*, September 1999.

[6] P. Resnik and N.A. Smith. The web as a parallel corpus. *Comput. Linguist.*, Vol. 29, No. 3, pp. 349–380, 2003.

[7] L. Shi, C. Niu, M. Zhou, and J. Gao. A DOM tree alignment model for mining parallel data from the web. In *Proceeding of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pp. 489–496, July 2006.