修　　士　　論　　文

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

Improving Efficiency of Parallel Text Extraction from Large Scale Text Corpora

指導教員　　**近山 隆 教授**

東京大学大学院新領域創成科学研究科
基盤情報学専攻

氏 名　　**66323 斎藤 大**

提 出 日　　**平成 20 年 1 月 29 日**

概 要

対訳テキストは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が，一般に十分な量・種類の対訳テキストを入手することは困難である．これまでに存在する対訳コーパスは限られた言語・ジャンルのものしか存在しておらず，人手で新しいコーパスを構築するには人的リソースや求められる規模の問題から困難である．そのため特定の言語・ジャンルの機械翻訳はある程度適切なものが出力される場合でも，汎用的なものを作ろうとするとリソース不足から急に精度が落ちてしまうといったことが起こりうる．

そこで近年，様々な言語・ジャンルのテキストが存在している Web から対訳テキストを抽出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，扱われているジャンルも多岐に渡っている．Web 文書で構築された大規模テキストコーパスから対訳ページを自動的に抽出することで巨大な対訳コーパスが構築出来ると期待される．

しかし，Web はテキストリソースとしては非常に巨大である反面，対訳でないページも多数存在するため，単純に大量のテキスト全ての組み合わせに対して対訳判定を行うのは現実的では無い．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．

そこで本論文では，Web から大規模なテキストコーパスを構築し高速に対訳テキストを抽出する手法を提案する．単純にはテキストコーパス上の全てのテキストの組み合わせに対し対訳テキストであるかどうかの全対全比較を行えば良いが，計算量のオーダが n^2 となってしまい，大規模化に対応するのは困難である．そこで，テキストコーパスから代表点となるテキストをランダムに選択し，各テキストをそれぞれ距離が近い代表点を持つクラスタに振り分けることで，対訳判定回数を削減する．また k-medoid 法との比較を行う．

次に速度向上のため LDA を用いた次元削減を提案する．多重トピックの文書生成モデルである LDA(Latent Dirichlet Allocation) を用いてトピック毎に重要な単語を抽出し，それらの単語を用いてクラスタリングを行うことで処理速度の向上を図る．クラスタリングにおいて精度をほとんど下げることなく，処理速度を 2 倍程度向上させることが出来た．

また転置インデクスを用いた高速化を提案する．検索エンジン等に使われる転置インデクスを対訳テキスト抽出の分野に応用することで計算量の削減を図る．クラスタリングの処理速度を単純な手法に比べ 6 倍程度向上させることが出来，LDA と組み合わせることで処理全体として 2 倍程度高速化出来た．

最後に大規模なテキスト群に対して提案手法を適用し大規模環境でも提案手法が有効なことを検証し，全対全比較に比べ 2.8 倍程度高速化出来た．

目 次

図 目 次

表 目 次

第1章　序論

1.1　背景と目的

対訳テキストは多言語自然言語処理分野の発展に欠かせない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た文章で意味内容が同じペア，いわゆる翻訳の形となっているテキストのペアを指す．対訳テキストを収集しデータ形式を統一したものは対訳コーパスと呼ばれ，様々な多言語自然言語処理の研究に用いられている．例えば統計的機械翻訳 [2, 4] では，大量の対訳テキストの統計データから単語の前後関係を基に翻訳されやすい意味を選択し機械翻訳を行う．対訳テキスト数の増加により翻訳信頼度が高められると期待される．また対訳辞書・シソーラスの構築 [17, 21] においては，対訳リソースの増大により辞書やシソーラスに含まれる語彙量や精度が向上することから，対訳テキストを大量に集めることで成果物の質の向上が期待される．他にも，対訳コーパスを利用した検索クエリ翻訳 [1, 8] や固有表現抽出 [24] 等が研究されている．

しかし，その重要性は十分に認識されていながらも，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る．そのため特定の言語・ジャンルの機械翻訳はある程度適切なものが出力される場合でも，汎用的なものを作ろうとするとリソース不足から急に精度が落ちてしまうといったことがありうる．この問題は様々な言語・ジャンルの大量の対訳リソースが存在すれば解決すると思われるが，これまでに存在する対訳コーパスは限られた言語・ジャンルのものしか存在していない．例えば対訳が提供されるジャンルの例としては

- 公用語が複数ある国や国連などの国際機関の公式文書
- 世界中で使用されるツールの説明書やヘルプ
- 翻訳辞書・外国語学習書の用例

等が挙げられる．また人手で新しいコーパスを構築するには各文書が対訳関係かどうかを逐一確認する必要があり，大規模な対訳コーパス構築は困難である．

そこで Web から対訳テキストを収集し対訳コーパスを構築する手法が提案されている [9, 19, 20]．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，扱われているジャンルも多岐に渡っている．特に童話やニュース記事など言語の壁を越えて世界中の人が興味を持つものに対しては多言語で記述・公開されているものが多数存在すると考えられ，Web 文書で構築された大規模テキストコーパスから対訳ページを自動的に抽出することで巨大な対訳コーパスが構築出来ると期待される．

しかし，Web はテキストリソースとしては非常に巨大である反面，対訳でないページも多数存在するため，単純に大量のテキスト全ての組み合わせに対して対訳判定を行うのは現実的では無い．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．

そこで本論文では，大規模テキストコーパスから高速に対訳テキストを抽出する手法を提案する．２テキスト間の対訳判定は，テキスト中の名詞を数列表現に変換しマッチングを行うことで高速化した手法 [10] を用いる．大量のテキストから対訳候補を高速に見つける手法として，クラスタリングによるテキスト振り分け・LDA を用いた次元削減・転置インデクスによる高速化 を組み合わせた手法を提案する．

1.2 本研究の位置づけ

本研究では，Web 上のテキストを中心とした大規模なテキストコーパスを構築し，そこから高速に対訳テキストを抽出することを最終的な目標としている．URL や HTML の構造を用いたテキストの対訳判定手法は提案されているが，大量の Web 文書に対してテキストの意味情報のみを用いた大規模な対訳コーパス構築はまだ行われていない．また本手法が用いた対訳判定手法 [10] では個々の対訳判定の高速さについては述べているが，大規模なテキスト集合に対する処理については全対全比較以上のことは言及されていない．

本手法ではテキストの意味情報のみを用いていることから，Web 上のテキストに限定せず既存の様々なテキストリソースから対訳テキストを抽出可能であると思われる．また，提案手法は対訳辞書と名詞抽出アルゴリズムさえ存在していれば様々な言語間で適用可能であるが，本研究の範囲では日英対訳テキストコーパスを用いた評価を行う．大規模化を考える上で必要となる，複数の計算機に並列に処理させることによる高速化も考慮に入れる．

1.3 本論文の構成

以下，本論文は次のような構成となる．

２章 関連研究
提案手法と関係している研究について述べる．特に高速化を目的とした個々のテキストペアの対訳判定について説明する．

３章 クラスタリングによる対訳テキスト抽出
大規模テキストコーパスから対訳を発見するための手法としてサンプリング及び k-medoid 法の説明と実験・比較を行う．

４章 LDA による対訳テキスト抽出
LDA(Latent Dirichlet Allocation) による次元削減の説明とそれに対する評価を行う．

5章 転置インデクスによる対訳テキスト抽出

主に検索エンジン等で使われている手法である転置インデクスを用いて，対訳テキスト検索の説明と実験，及びそれを応用した対訳テキスト抽出の高速化手法の説明と実験を行う．

6章 大規模データへの適用

提案手法を実際に Web から収集したテキスト群に対して実行し，その評価を行う．

7章 結論

結論と今後の課題を述べる．

第2章　関連研究

本章では関連研究として，2.1 において Web から対訳テキストを抽出した例について紹介する．次に 2.2 において大量の要素に対するクラスタリング手法について紹介する．また 2.3 において LDA(Latent Dirichlet Allocation) について説明し，最後に 2.4 において本手法で用いた高速な対訳判定手法について説明する．

2.1　対訳テキスト抽出

本節では対訳テキスト抽出の既存研究について説明する．特に大量にテキストが存在する Web 空間から，対訳ペアを抽出するために行われている手法を述べる．以下

- URL マッチング
- リンクマッチング
- HTML 構造マッチング

の 3 手法について説明する．

2.1.1　URL マッチング

URL マッチングは，対訳関係のテキストは URL も大部分が重なるはずであるという前提を用いた手法である．例えば

$$http://www.hostname.co.jp/index.html.ja$$

$$http://www.hostname.com/index.html.en$$

上の 2 つの URL のように，一部を除いて URL が一致した時のみテキストの中身を詳細に見て対訳であるかどうかを判定する．

Resnik ら [19] は，Internet Archive[1]を対象に対訳テキストの収集を行った．この手法では Web ページ群から対訳の可能性が高いテキストを発見するために URL マッチングを用いている．URL が似ているテキストペアは対訳となっている可能性が高いとの判断から各ページの URL に対して書かれている言語を示唆する文字列 (LSSs: Language Specific Substrings) (例えば日本語の場合は ja,

[1] http://www.archive.org

jp, japanese, shift_jis 等) を除去し，除去後の文字列が一致したページ群のみ対訳を判定することで大幅に計算コストを削減している．URL 一覧は，検索エンジンを用いて同一ページ内の anchor タグに ‘English’ と ‘Japanese’ が両方含まれるページを取得，あるいは InternetArchive から取得してきている．また URL マッチング後の対訳判定にはテキスト中の対訳語の出現位置比較と HTML ツリーの類似性を用いている．これにより 1900 万ページから 8294 ペアの対訳候補を発見し最終的に 2190 ペアの英語－アラビア語の対訳テキストを取得している．

我々が保持している URL データを用いて実際にどの程度の対訳 URL が抽出されるかの実験を行った．実験データは我々の研究室が開発したクローラである Shim-Crawler[2]を用いて，Open Directory Project の大学関連ニュースページ[3] を seed としてページ収集→リンク抽出の繰り返しを 12 回行い収集したおよそ 9000 万 URL である．日英の URL マッチングを行うため，日本語は (japanese, .ja, _ja, ja., ja_)，英語は (english, .en, _en, en., en_) のそれぞれ 5 パターンを LSSs として使用した．結果として，9000 万 URL からおよそ 7,400URL・3,700 ペアを収集することが出来た．Resnik らが示した結果よりも取得されたペアが少ないが，これは LSSs の設定や収集したページのスコープの違い・言語に依存するものが原因であると思われる．また実際抽出された URL のうちいくつかを見てみた結果，画像がほとんどで対訳となりうる文章がほぼ含まれていない企業のサイトのようなページが見られた．実際全ページのうちどの程度の対訳ペアが含まれているのかは定かでは無いが，例えばニュース記事のような URL マッチングが適用できない対訳ペア例が挙げられることから URL マッチングが利用できる範囲が限られてしまい，この条件に合わない多くの対訳ページを見逃している可能性がある．

2.1.2 リンクマッチング

リンクマッチングは HTML のリンク構造を利用し，ほぼ確実に対訳が存在するであろう Web サイトを見つけ，それに対して人手でルールを記述することで大量の対訳テキストを収集する手法である．例えば複数言語に対応しているニュースサイトの場合，日本語記事から英語記事へのリンクが存在したら元テキストとリンク先テキストはほぼ確実に対訳となる．

Fry[9] は，あらかじめ対訳テキストを豊富に含んでいるサイトを対象に，そのサイトのリンク構造・HTML 構造を利用して気軽に対訳テキストを収集できる手法を提案した．実装は簡単なマッチングルールを作るだけであり，収集できる対訳テキストの質・量・増加速度などが予測可能であるというメリットもある．英文ニュース記事を和訳して配信するニュースサイトの RSS を用いて 17,277 の対訳ペアを発見しており，その結果は日英対訳コーパスとして Web 上で公開されている．しかし実際にはテキスト内部の対訳判定は行っていないので，公開されているコーパスを見るとある程度の対訳ではないペアが含まれている．これは記事自体に大胆な省略など大幅な構成の変更が為されていたり，全く関係の無い記事へのリンクを誤って検出してしまっている可能性が考えられる．

[2] http://www.logos.ic.i.u-tokyo.ac.jp/crawler/
[3] http://www.dmoz.org/News/Colleges_and_Universities

2.1.3 HTML 構造マッチング

HTML 構造マッチングは，対訳ページ中のタグ情報を用いて，title タグや hn タグの文字列が一致した場合に重みをつけたり，同じようにリンクが張られているリンク先を優先して対訳判定することで無駄な計算コストを削減した手法である．

Lei ら [20] は，DocumentObjectModel(DOM) と呼ばれる HTML 構造をツリー状に表現するモデルを用いて効率的な対訳テキスト収集を行っている．対訳テキスト中から同じようなリンクが張られていた場合はそのリンク先のペアも対訳の可能性が高いとの仮定の下で，対訳ペアの同じような形のリンク先をほぼ対訳であるとみなして判定を行うことで，無駄な対訳判定を減らしている．最初に対訳を見つけるために，各ホストのトップページ (index.html) とそこからリンクが張られているページを集めてきて，タグの alt やリンクの文字列に例えば'in English' や'English version' のような言語を表す言葉が入っている場合に対訳判定を行う．実際の対訳の判定にはテキスト長による重み付けや HTML タグ構造・単語の意味比較などを用いている．これにより URL をベースとした対訳判定に比べて同等の対訳ペアを取得していながら，およそ 2.26 倍の取得対訳ペア数対取得ページ数の効果が得られ，少ないページダウンロード数で対訳が収集できることを示している．ただし問題点として取得できる Web ページの形式が限られること，DOM ツリー構築にはコストがかかること等が挙げられる．

2.1.4 まとめ

Web から対訳テキストを収集する場合，大きく分けて

- 対訳テキスト候補の発見
- テキストペアの対訳判定

の 2 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 [5, 9, 18, 19]，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 [13, 20] 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 [13] や URL 文字列・HTML ツリーの類似性を用いた判定 [5, 19]，HTML 構造を用いた判定 [20] 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．

我々はこれらに対して，探索範囲が広く大量のテキストを扱う場合でも計算量が削減されるような手法を提案する．

2.2 大規模クラスタリング

本節では大量のデータに対するクラスタリング手法について説明する．

k-means 法

まず，データクラスタリング手法の代表的なものとして k-means 法がある．これは実際一般的に最もよく用いられているクラスタリング手法であり，[14] で提案されている．k-means 法は分割最適化手法とも呼ばれ，クラスタ数 k を与えた時に各クラスタの平均を求めながら以下のようにクラスタリングを行う．

1. 全データからランダムに k 個のデータを選びそれぞれのクラスタの代表点とする
2. 全データを最も近い代表点を持つクラスタに分類する
3. 各クラスタにおいて平均を計算しなおし代表点とする
4. 全てのクラスタの代表点が変化しなくなるまで 2,3 を繰り返す

k-means 法はランダムに選択するクラスタの代表点の初期値により結果が大きく変わってくる．そこで，いくつかの異なる初期値から初めて得られた結果から最も良いものを選ぶ必要がある．

k-medoid 法

次に紹介する手法は k-medoid 法である．k-medoid 法は別名 PAM (Partition Around Medoids) とも呼ばれ，k 個のクラスタを $C_1, C_2, ...C_k$ とした時各クラスタの代表点 $o_1, o_2, ...o_k$ を選ぶ．k-means 法とはクラスタの代表をクラスタ内データの平均で取らず，あくまで要素の 1 つを代表点として用いている点が異なる．N 個の各データ $d_1, d_2, ...d_i, ...d_N$ を，$\min_{1\leq j\leq k} dist(d_i, o_j)$ を満たす o_j，つまり最も距離が近い medoid が所属するクラスタ C_j に分類する．k-medoid 法は medoid の選択が非常に重要な問題であるが，初期の medoid はランダムに選択し，以降クラスタの振り分けと medoid の選択を交互に行っていくことで適切な medoid の選択を逐次的に行っている．クラスタリングによって割り当てられたデータと medoid との距離の総和

$$\sum_{i=1}^{N} \min_{1\leq j\leq k} dist(d_i, o_j)$$

をポテンシャルと呼び，クラスタリングの評価としてポテンシャルが最小になるようなデータの振り分けを行う．k-medoid 法は単純であるが，計算量が $O(N^2k)$ と大きくなってしまうという問題点がある．k-means 法と k-medoid 法の概念的な違いを図 2.1 に示す．

CLARA

k-medoid 法の計算量における欠点を克服し大規模データに対応した手法として CLARA (Clustering LARge Applications)[12] が提案されている．この手法はランダムサンプルを用いていて，まずデータ全体から $2k + 40$ 個のランダムサンプルを取得し，その中で k-medoid 法を適用して medoid を選択する．その後ランダムサンプルされなかったデータを各 medoid に対して割り当てることでクラスタリングを行う．この手続きを何度か実行して最もポテンシャルが

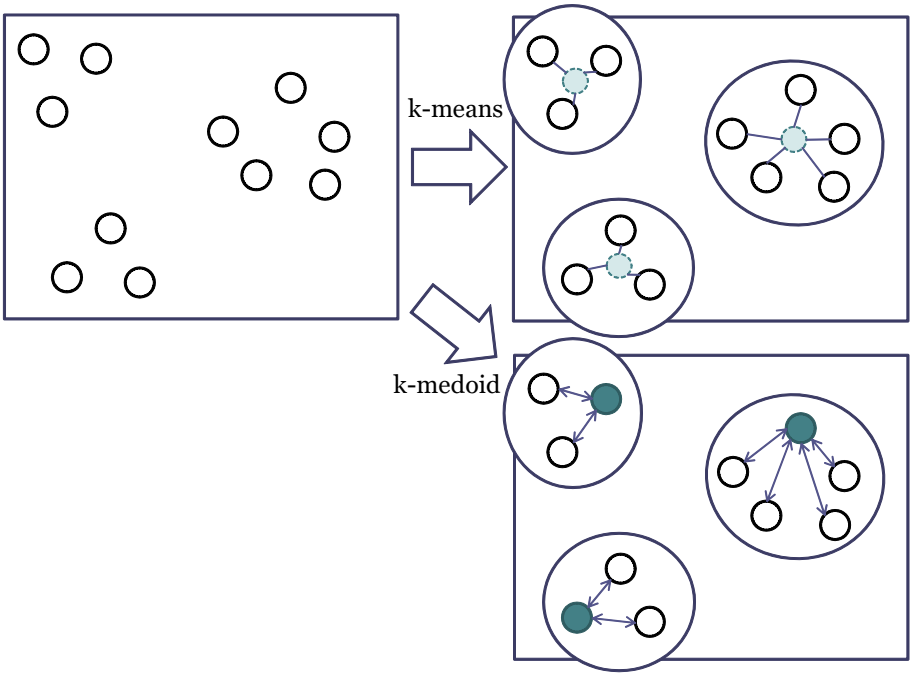

図 2.1: K-means 法と K-medoid 法の概要

小さくなるクラスタリングを出力する．しかし，この手法は初めのランダムサンプルに精度が大きく依存し，クラスタリングの精度が保障されない問題点がある．

CLARANS

CLARA のクラスタリング精度を k-medoid 法に近づけるための手法が CLARANS (Clustering Large Applications based on RANdomized Search)[15, 16] である．CLARA が最初にランダムサンプルを選ぶのに対し，CLARANS では k-medoid 法の medoid を選択するところでランダムサンプリングを行い，そのサンプル内のみで新たな medoid の評価を行う．[15] では，経験的に $k(N-k)/80$ と 250 の大きい方を選択すると良いとされている．実験結果によると，同じ時間内では CLARANS の方が CLARA よりもポテンシャルの小さいクラスタが出力されている．

2.3 Latent Dirichlet Allocation

LDA(Latent Dirichlet Allocation) [3] は，多重トピック文書の生成過程を 3 層の階層型ベイズモデルによりモデル化した文書生成モデルである．文書はいくつかのトピックをある割合で持ってい

てトピックから単語が確率的に生成されるという考えの下で，文書は各トピックが確率的に生成した単語から構成されるというモデル化を行っている．

LDA による文書生成の流れについて説明する．文書集合におけるトピック数 K が与えられているものとし，出現する単語の種類数を V とする．基本的な流れを以下に示す．

1. Poisson 分布に従い文書の単語数 N を選択
2. Dirichlet 分布 $Dir(\alpha)$ に従い文書が持つトピックの重み割合ベクトル θ を選択
3. 単語数 $(n = 1, 2, ..., N)$ だけ以下繰り返し
   (a) 多項分布 $M(\theta)$ に従いトピック z_n を選択
   (b) トピック z_n と β から単語 w_n を生成

ここで α は各トピックからの単語生成の Dirichlet 分布パラメータ $(\alpha = \alpha_1, ..., \alpha_K)$ であり，β は各トピックから各単語が生成される確率 $(\beta = \beta_{11}, ..., \beta_{KV})$ である．

まず，生成文書毎に α を Dirichlet 分布パラメータとしてトピックの重み割合ベクトル $\theta(\theta = \theta_1, ..., \theta_K)$ を以下のような確率密度関数から求める．

$$p(\theta|\alpha) = \frac{\Gamma(\sum_{i=1}^{K} \alpha_i)}{\prod_{i=1}^{K} \Gamma(\alpha_i)} \prod_{i=1}^{K} \theta_i^{\alpha_i - 1}$$

この式は「各トピック i の単語生成が $Dirichlet(\alpha_i)$ の分布に従うとき，トピックの重み割合 θ を持つ文書の生成確率密度」という意味である．

次に，n 番目の単語のトピック z_n を θ をパラメータとする多項分布 $M(\theta)$ に従い選択し，z_n と各トピックにおける各単語の生成確率 β から単語 w_n を生成する．以上を定式化すると以下のようになる．

$$p(\theta, z, w|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta) p(w_n|z_n, \beta)$$

ここで z は N 個のトピック $(z = z_1, ..., z_N)$，w は N 個の単語 $(w = w_1, ..., w_N)$ である．つまり，α と β が与えられることで，トピックの重み割合ベクトル θ, N 個のトピック z, N 個の単語 w が求められることになる．文書 (=単語の集合) の周辺分布は以下のようになる．

$$p(w|\alpha, \beta) = \int p(\theta|\alpha) (\prod_{n=1}^{N} \sum_{z_n} p(z_n|\theta) p(w_n|z_n, \beta)) d\theta$$

以上のように文書を生成する．実際に文書生成を行う場合は，文書集合からこの α と β を EM アルゴリズムを用いて推定する．

LDA の特徴として，以下のような点が挙げられる．

- bag-of-words モデル
- 多重トピック文書に対応

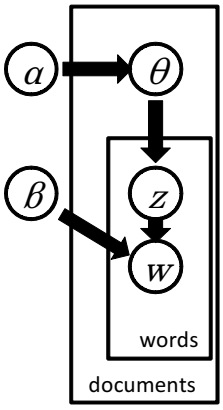

図 2.2: LDA のグラフィカルモデル

- トピック数は事前与える必要有り

LDA のグラフィカルモデルを図 2.2 に示す．外部パラメータ α と β が存在し，α により θ が求められる 1 層・θ により z が求められる 2 層・z と β により w が求められる 3 層と分かれているのが見てとれる．一般的な 2 層の文書生成モデルは文書のトピック選択とトピックからの単語生成という 2 つに分かれているが，LDA ではトピックの重み割合ベクトルを求める層を追加することで多重トピック文書に対応出来ている．

2.4 対訳判定

対訳判定とは，2 つのテキストが与えられた時にそれらが実際に対訳であるかどうかを判定するものである．集められる対訳コーパスの精度はこの対訳判定手法の精度に大きく依存するため正確な手法を用いる必要があるが，大量の対訳判定を行うためには計算コストが低く高速に実行可能という性質も重要である．素朴な対訳判定方法として，2 つのテキストに出現する単語の全ての組み合わせについて対訳関係があるかどうかを辞書を参照しつつ数え上げることが考えられるが，この場合の計算量はテキスト長の 2 乗と辞書の検索コストの積に比例し非常に大きなものとなる．そこで対訳辞書を用いてノードを各単語・エッジを対訳関係としたグラフ構造を構築し，異なる言語で記述されたテキストを数列に変換することで対訳判定の高速化を図る手法 [10] を用いる．この手法の基本的な流れは

- 対訳辞書からグラフ構造を構築
- テキストを意味 ID の数列に変換
- 各テキストを比較

となる．流れのイメージ図は図 2.3 のようになる．それぞれについて説明する．

2.4.1 グラフ構築

対訳辞書からグラフ構造を構築する手法について説明する．まず対訳辞書は NICT から提供されている EDR 電子化辞書 [7] を用いた．辞書には日本語→英語の翻訳と英語→日本語の翻訳・品詞や語の意味・同意語などが載っているが，グラフ構築を行うために品詞と対訳関係のみを用いた．‘言語 L_1 の単語 W_i^1 が言語 L_2 の単語 W_j^2 に翻訳可能である’ ことを ‘対訳関係 T_{ij} が成立する’ と定義すると，対訳辞書中の各単語 $W_1^1, W_2^1, ..., W_1^2, W_2^2, ...$ をノード，各対訳関係 $T_{i_1j_1}, T_{i_2j_2}, ...$ をエッジとみなすことで辞書情報をグラフ構造として扱うことが出来る．各ノードは L_1 あるいは L_2 に属する．一般に対訳関係 T_{ij} が辞書中に存在しても T_{ji} が存在するとは限らないのでこのグラフは有向グラフであるが，しかし後に述べる手法ではこのグラフを無向グラフとして扱いたいので対応する T_{ji} が存在しない T_{ij} はあらかじめ取り除いておく．つまり T_{ij} と T_{ji} は同値である．辞書情報をグラフ構造として扱い，各連結成分に対してインデックスとしてそれぞれ意味 ID を与える．

多くの単語は複数の訳が存在する．そのため対訳関係が存在する単語を次々と接続していくと図 2.4 のように意味的に全く関係の無い単語が結果として接続されてしまうことがある．しかし実際に EDR 電子化辞書からグラフ関係を構築したところ任意の 2 単語間にそのような経路が存在するわけではなく，グラフは複数の連結成分から構築されていることが確認できた．ただし非常に大きい連結成分が存在する可能性があり，例えば辞書に登場する多義語のほとんどを包含するような巨大な対訳グラフが構築される事態が起こりうる．このような連結成分が存在した場合，対訳判定の品質を著しく下がることが予想される．そのための改善手法として大きな成分を無視して判定を行う方法もあるが，巨大な成分には多くの多義語を持つ語が含まれる可能性が高く，一般にそのような語は頻繁に用いられる傾向にあるのでその情報を出来る限り利用したい．そこでこの手法では対訳グラフ中の巨大成分の分割を行っている．具体的な分割のアルゴリズムは省略するが，基本的にはランダムに初期値を与えて山登り法を用いて最適なグラフ分割が行われている．Fry の Web コーパス [9] に対して様々な分割方法を適用したグラフを用いたところ，各連結成分に含まれる全ノード数が 30 ノードになるまで分割し，3 桁までの数詞情報を追加したものが最も良い性能を示したグラフである，と示されている．最適なグラフは対訳判定を行うテキストの種類によって様々であると考えられるが，今回はこのグラフを用いて実験および性能比較を行う．

2.4.2 テキストの数列変換

対訳判定を行うためには英語・日本語それぞれに対して形態素解析を行い単語分割・品詞判定・原型抽出などを行う必要がある．形態素解析プログラムはこれまでも広く研究されており，様々なライブラリが公開されている．英語に関しては単語分割・品詞判定に SS Tagger[4]，原型抽出に WordNet[5]を用いている．日本語に関しては有名な形態素解析プログラムとして MeCab[6]があり，それを用いている．我々の提案する手法で Web 文書を意味 ID に変換する場合もこれに倣う．

[4] http://www-tsujii.is.s.u-tokyo.ac.jp/~tsuruoka/postagger/
[5] http://wordnet.princeton.edu/
[6] http://mecab.sourceforge.jp/

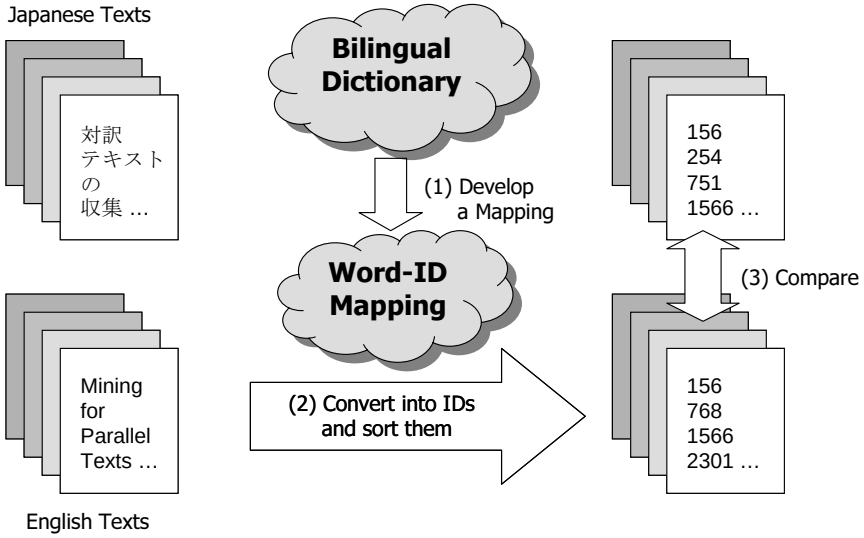

図 2.3: 対訳テキスト判定の流れ

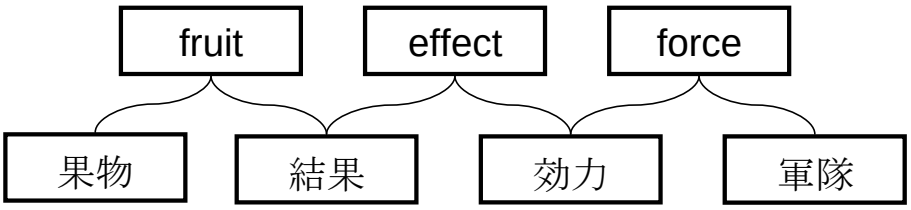

図 2.4: 対訳辞書による単語の連結

形態素解析後のテキスト中の各単語に対して辞書照合を行い，辞書中に存在する名詞のみを取り出して意味 ID に変換する．つまり，先に生成された各連結成分にユニークな ID が振られた対訳グラフを用いて，テキスト中の各単語について辞書に存在するかどうかをチェックし，存在したらその単語を対応する対訳グラフの ID（意味 ID）で置き換える．テキストの数列変換の概要を図 2.5 に示す．対訳グラフを無向グラフとしたことで，１つの連結成分中の単語全てを同じ ID で置き換えることが出来る．辞書に存在しない単語は無視する．また各単語のテキスト全体における正規化された位置情報 (文の先頭が 0.0，末尾が 1.0) も保持する．これは同じ意味を持つ単語であっても例えばテキストの最初と最後など離れた位置に存在していた場合それらは対訳単語とみなさない方が精度が上がることが予想されるためである．これにより，テキスト中の単語を

$$(\text{意味 } ID, \text{出現位置})$$

の２次元ベクトルの形に変換する．この結果得られた数値集合について，意味 ID・出現位置の順にソートしてテキストの数列への変換は完了する．これでいくらか削られてはいるものの各テキストの情報が数列で表現されていることになる．

2.4.3 テキスト比較

数列表現された２つのテキストを比較する方法について説明する．基本的にはソートされた数列について要素の先頭から順に比較を行う．それぞれの数列の先頭要素にカーソルをセットする．カーソル位置の２つの要素が同じ意味 ID を持っていて単語位置の差も基準値以内の場合，単語の対訳関係が見つかったとして対訳カウントを１増やし，両方のカーソルを１つ進める．意味 ID が一致しない場合は番号が若い方のカーソルのみ１つ進める．どちらかの数列が終端に行くまでこの比較を繰り返し，見つかった対訳カウントの２つの数列長に対する比率から対訳らしさの判定を行う．[10] ではこの値を $tscore$(translational score) として定義されているが，この名前は２単語の共起関係の指標である $tscore$[6] と重なってしまうので，本論文では trans_score と呼ぶ．trans_score の定義は以下のようになる．

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

計算コストの面では１回の比較で少なくとも１つのカーソルが進むので，比較回数は最大でも２つの数列の長さの和となる．

対訳判定時のパラメータとして

- 単語位置の threshold
- trans_score の threshold

の２つを与える必要がある．単語位置の threshold は 2 単語 (ここでは意味 ID) 間の距離がこの値以下の場合に対訳単語とみなす基準となる値で，高くしすぎると位置が離れている単語でも対訳単語とみなしてしまい，低くしすぎると本当の対訳単語を見逃す可能性がある．同じように trans_score の threshold はこの値以上を対訳とみなすというしきい値であり，これについても高くしすぎると実

際は対訳でないペアを対訳とみなしてしまい，低くしすぎると実際の対訳ペアも見逃す可能性がある．[10] では fry の 200 ペアを用いて，距離の $threshold = 0.2$，trans_score の $threshold = 0.102$ とすることで最も良い結果が出た，と示されている．ここで結果が良いとは

$$F = \frac{precision \times recall}{precision + recall} \times 2$$

で定義される F 値が最大となるパラメータのことを表している．$precision$ は適合率と呼ばれ対訳と判定した内実際に対訳となっているものの割合，$recall$ は再現率と呼ばれ対訳と判定したものが全体の対訳数の占める割合である．図 2.6 に示されるように一般にこの 2 つの値はトレードオフの関係にあり，F 値が最大となる点 (この場合は $F = 0.982$) が最も良い場合であると言える．今後，この threshold を用いて対訳判定の高速化を行う．

また [25] では大量のデータから対訳テキストを全対全で比較した場合の精度についても言及している．trans_score はデータの量が増えると精度が下がることも実験で示されていて，Fry のコーパス 200 ペアに対しては 0.982 と高い F 値となっているが，10000 程度の非対訳テキストに 400 ペアの対訳テキストを混ぜて全対全比較で対訳判定する実験を行った場合では F 値が 0.790・recall が 0.680 程度であったと報告されている．またその原因の 1 つとして，Fry のコーパスに載っているニュース記事自体に大きく形が変わる意訳等が施されている等により，1 〜 2 割程度は本来対訳で無いものが対訳であるとして提供されていることを挙げている．このように大量のデータの中から対訳テキストを発見する際には trans_score が 100%信頼出来るわけではないが，本研究の範囲では後に述べるクラスタリングによって本来対訳であるテキストが同じクラスタに入ることを目的とし，振り分け後の個々の対訳判定の精度は今後の課題としたい．

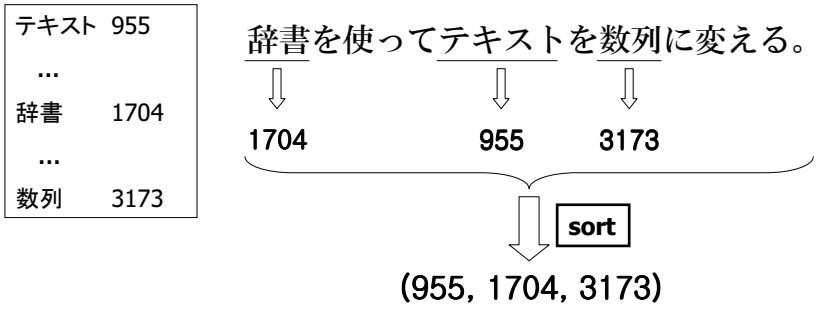

図 2.5: テキストの意味 ID 変換

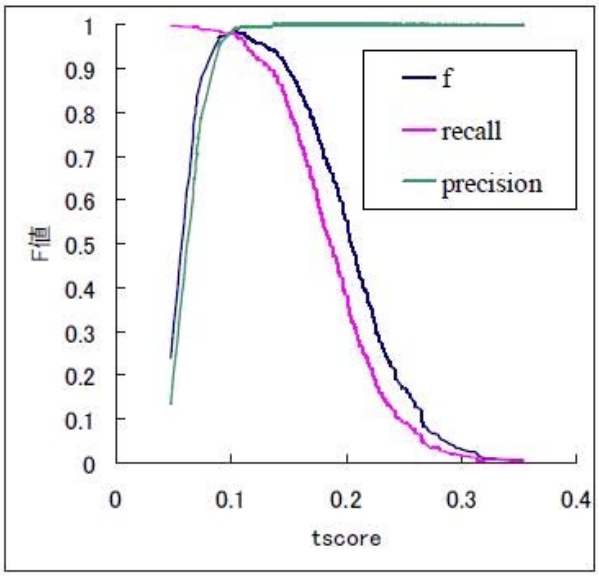

図 2.6: 200 ペアの日英対訳テキストを対象とした Trans_score の F 値

第3章　クラスタリングによる対訳テキスト抽出

2.4 で説明した手法は個々の対訳判定の高速化を狙ったものである．しかし実際に大規模テキストコーパスからの対訳テキスト抽出を考えた場合，単純に全対全の比較を行ってしまうと計算量がオーダ n^2 となってしまい，たとえ個々の対訳判定が高速化されていたとしても現実的な時間で終わらなくなってしまう．大規模な対訳テキストコーパス構築を目指すには，対訳判定時間を減らす他に対訳判定の回数を減らす方面からのアプローチも考える必要がある．判定回数を減らすと言ってもその作業によって実際に対訳であるペアの判定まで削られたら意味が無いので，低コストで明らかに対訳で無いものを弾くような処理が必要となる．

そこで提案手法では，高速に対訳テキスト判定が可能な trans_score を用いてクラスタリングを行うことで大量のページの対訳判定を行う方法を提案する．URL フィルタリングのように対象となるページの範囲が厳しく制限されることもなく，辞書の単語空間をそのまま用いている手法よりも計算コストが大幅に下がることが期待される．

以降，本章ではクラスタリング手法であるサンプリングと k-medoid 法について説明し，両者の評価を行う．

3.1　サンプリングを用いた高速化

本手法ではサンプリングを用いた計算コスト削減を行う．テキスト集合の中から代表点となるテキストをサンプリングし，代表点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距離が近い代表点を持つクラスタ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 と比べて小さい数の代表点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代表点を持つクラスタに振り分ける．理想的には最も距離が近い代表点を持つクラスタ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クラスタに振り分けられる危険を避けるための対処として，距離が近い代表点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 3.1 に示す．また，手順を以下に示す．

1. サンプルテキストをランダムに選択
2. 全テキストとサンプルテキストとの距離を計算
3. 最も近いサンプルテキストのクラスタに振り分け
4. クラスタ内で全対全の対訳判定

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．

本手法の計算コストに関する考察を行う．テキスト数が n の時単純に全対全の比較を行った場合，計算コストのオーダは n^2 となる．ここでテキストを m 個の集合に均等に分割してその集合内でのみ全対全比較を行うと，n/m 個の要素を持つクラスタ内での全対全比較が m 回行われ，更に分割で n 個のテキストと m 個の代表点との距離を測定するので，全体のコストは

$$(n/m)^2 \times m + nm = n^2/m + nm$$

となる．またこの式の理論的な最小値は

$$n^2/m + nm \geq 2n^{3/2}$$

より，$m = \sqrt{n}$ の時に $2n^{3/2}$ で最小となる．つまり，各テキストの集合が均等に分割されたとすると，理論上分割数をテキスト数の平方根とすることで最も計算量が削減できることとなる．このように巨大な集合を小さな集合に分割してから対訳判定を行うことでテキストの比較回数が削減出来る．この手法の場合特に計算量がかかるフェーズとして

1. 全テキストとサンプルテキストとの距離を計算して振り分け
2. 振り分けられたクラスタ内での全対全比較

の 2 つが考えられる．

3.2 K-medoid 法を用いた高速化

サンプリングの単純な拡張として，k-medoid 法を用いたクラスタリングが考えられる．サンプリングは代表点をランダムに選ぶためその選択によって結果が左右される可能性があるが，k-medoid 法を用いると適当な代表点から開始しても最終的に得られる結果が安定することが期待される．ただし収束するまで何度も繰り返し計算を行うため計算コストが大きくなってしまい大規模データの扱いが困難となる問題点がある．

クラスタ数 k とした時の k-medoid 法の流れは，

1. クラスタの中心をランダムに設定する．
2. それぞれを最も近い中心に割り当てる．
3. クラスタごとに中心を計算し，選択しなおす．
4. クラスタ中心の変化が収束すれば終了．それ以外は 2 へ戻る．

となる．以下，特にサンプリングと異なる点であるクラスタ中心の選択及び終了条件について説明する．

クラスタ中心の選択方法について述べる．各テキストがそれぞれクラスタに振り分けられたとき，クラスタ毎に単語出現数の平均値を定める．我々の手法では，単語として意味 ID を使用しているのでクラスタ毎に意味 ID の出現回数の平均を取る．また距離の尺度として trans_score を使用しており，trans_score を計算するためには意味 ID 列が整数である必要があるので，クラスタ毎の意味 ID の出現回数の平均値に対して小数第 1 位で四捨五入を行うことでクラスタ内テキストの平均を取った意味 ID 列を用意することが出来る．この平均とクラスタ内の各テキストとの距離を trans_score で計算し，最も trans_score が大きい（＝距離が近い）テキストがそのクラスタの新たな中心となる．

次に，終了条件について述べる．通常はクラスタ中心の変化が収束した時を終了条件とすれば良いのであるが，k-medoid 法の問題点の 1 つとして，中心選択がループに陥る可能性がある．そのため，終了条件は中心の変化が起こらなくなった場合だけではなく，中心の変化が繰り返しに陥った場合も加える必要がある．

3.3 並列化

クラスタリングの利点となる性質の 1 つとして並列化が可能であることが挙げられる．はじめのテキストを振り分ける段階では全てのテキストを見る必要があるが，振り分け後はそのクラスタ内のテキストしか見る必要が無くなるので単純に並列化を施すことが可能である．具体的には代表点となるサンプルテキストを選択した段階で各代表点を持つクラスタにそれぞれ担当のホストを決めておくことでデータを各ホストに分散させ振り分け後の全対全比較は各ホスト上で行うことが可能となる．

3.4 実験

3.4.1 実験準備

3.4.1.1 多言語テキストの扱い

対訳テキストを扱うためには，複数の言語で記述されたテキストに対する処理を考える必要がある．通常，一方の言語のテキストから他方の言語のテキストへの変換は対訳辞書を用いて単語レベルで行われるが，辞書には複数の訳が存在するためその変換自体容易ではない．本手法では 2.4.2 で用いられた変換から

1. 対訳辞書からグラフ構築
2. 名詞抽出
3. 意味 ID 変換

の流れで対訳判定に必要な情報を残しつつ，複数言語で記述されたテキストを意味 ID 空間という限られた次元の数列へと落とし込む．そのため言語によらず各テキストに対して同じ処理を施すことが出来，また各テキストに言語情報を表すラベルを付与することで最終的な対訳判定の時のみ言語情報を利用する，といったことが可能となる．

3.4.1.2 Web テキストの扱い

2.4 の判定方法は自然言語に対応したものである．しかし Web テキストは自然言語のまま記述されていることは少なく，HTML(Hyper Text Markup Language) で書かれているものが多数を占める．そこでまずは HTML 文書を自然言語に変換する必要がある．単純には HTML タグ (<> で囲まれているタグ) を取り除くだけで対応可能だが，例えば title タグや hn タグを重視するといったような HTML のタグ情報を使おうとすると HTML の構造解析から行う必要がある．またニュースサイトやブログのように本来必要な記事がページ内の一部にしか存在しない場合は，繰り返し出現する部分を判定し HTML から削除する手法 [22] も提案されている．本手法では正規表現によりタグを除去することで HTML から自然言語への変換を実装している．

また Web 上で使用されている言語は複数存在するため，言語判定を行う必要がある．また日本語に関しては文字コードの問題もあり，MeCab は EUC-JP にしか対応していないため文字コード変換も必要となる．本手法では言語判定のために N-Gram を用いた言語判定器であり Perl モジュールとして配布されている Lingua::LanguageGuesser[1] を用いた．このモジュールの特徴として UTF-8 に対応していることが挙げられ，多くの日本語文書に対応することが出来る．これにより判定された言語・文字コードに従い，nkf コマンドで文字コード変換を行う．

3.4.1.3 評価に扱うデータ

評価に用いるため，Fry が公開している対訳 URL[9] に対して我々が扱いやすい形に変換を行う．公開されている情報は URL の一覧だけなので実際にはその URL にアクセスして HTML を取得し，3.4.1.2 で説明した手法を用いて自然言語レベルまで変換する必要がある．このコーパスは３・４種類のニュースサイトから記事を集めているだけなので，サイト毎にヒューリスティックにルールを定めて各記事の本文抽出を行った．

[1] http://gensen.dl.itc.u-tokyo.ac.jp/LanguageGuesser/LanguageGuesser_ja.html

3.4.1.4 距離尺度の評価

我々の提案手法では距離の尺度として 2.4 で述べた trans_score を用いる．ただし文書，ここで言う意味 ID 列をベクトルとして見た場合のベクトル間距離については他にも様々な距離尺度が考えられる．そこで距離尺度としての trans_score の妥当性について検証するための予備実験を行う．

他に考えられるベクトル間距離として，以下のようなものが挙げられる．

AND

2つの数列間で同じ意味 ID が存在した場合に score を 1 加算する．出現単語が同じ場合に score が高くなる．

NOT XOR

2つの数列間で同じ意味 ID がどちらも存在する，あるいはどちらも存在しない場合に score を 1 加算する．異なる出現単語が少ない場合に score が高くなる．

AND - XOR

2つの数列間で同じ意味 ID が存在した場合に score を 1 加算し，異なる意味 ID が存在した場合に score を 1 減算する．AND に誤り単語の評価を追加した形．

EUCLID

各ベクトル間のユークリッド距離．$\sqrt{\sum(x_i - y_i)^2}$.

COS

各ベクトル間のコサイン距離．$x \cdot y/|x||y|$.

対訳コーパスに対してそれぞれの距離尺度を用いた場合に本来の対訳がどの程度近くなるか，で評価を行う．実験対象とするテキストは Fry の WebCorpus から抽出した 200 の日英対訳ペアである．それぞれの距離尺度を用いて日本語×英語の距離計算を 200 × 200 の 40,000 回行い，実際に対訳となっているものがどの程度近いかを測定した．日本語→英語を探索する場合と英語→日本語を探索する場合があるので，日英合わせて 400 テキストを元に距離を測定することになる．

結果を表 3.1 に示す．表の各数値は指定した距離尺度を用いて各テキストについて他のテキストとの距離を測定した時に，その対訳が指定した Rank 内に入っている数である．最も適切な距離尺度を用いた場合，この数値は 400 となり，400 に近いほど距離尺度の性能が高い，と言える．これによるとやはり trans_score が 2 つの対訳関係を表す指標としては適切であることが示されている．また 5 つのここで定義した距離の中では AND が最も適切な結果を示しているが，trans_score の方が精度が良いのは明らかである．

以上より，本手法では距離尺度として trans_score を用いる．

3.4.2 実験概要

評価用のデータとして Fry 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．

(/400)	Rank 1	Rank 3
AND	368	389
NOT XOR	53	212
AND - XOR	336	391
EUCLID	126	209
COS	307	368
trans_score	398	398

表 3.1: 各距離尺度による指定したランク内での対訳テキスト抽出数

まずは振り分け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

- 距離が最も近いサンプルに振り分け (多重度=1)
- 距離が近い順に 3 サンプルに振り分け (多重度=3)

の 2 つである．対訳判定はペア数を 200, 400, 800, 1600, 3200, 6400, 12800 と増やして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 2 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．振り分けでは同じサンプルに振り分けられたテキスト間でのみ全対全で対訳判定を行い，振り分けられていないテキストとの比較を行わない．ここでのサンプル数は $\sqrt{n}$ とした．これにより，振り分けによる計算速度向上の評価を行う．

次に，サンプル数を変化させたときの計算コストと精度への影響を測定した．Fry の WebCorpus200 ペアに対して，サンプル数を 1〜28($=int(\sqrt{200}) \times 2$) の範囲で変えて行った時の単語比較数及び誤分類率を測定した．ここで誤分類率とは本来対訳であるペアが別のクラスタに振り分けられて対訳が発見できない確率であり，(対訳が発見出来なかった数)/200 で表される．この測定はサンプリングと k-medoid 法の両方で行いこの 2 手法の比較を行った．

次に，多重度と計算コスト・誤分類率の関係及び計算コストと誤分類率の関係を測定した．同じく Fry の WebCorpus200 ペアに対して，距離が近いサンプルのうちいくつに振り分けるのか (=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及び誤分類率を測定した．多重度を固定した時のサンプル数=14 の単語比較数及び誤分類率をグラフで示す．また，今回測定した全ての多重度・サンプル数での単語比較数と誤分類率の関係をプロットしたものを示す．更に，並列化による計算コスト削減の効果を測定するため，振り分け後の各クラスタにおける全対全の対訳判定を並列化させた場合の単語比較数と誤分類率の関係もプロットする．これらもサンプリングと k-medoid 法の両方で行う．

3.4.3 実験結果

3.4.3.1 振り分けによる計算コスト削減

まず，振り分けによる計算コスト削減の結果を図 3.2 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．また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も見て取れる．このように処理速度だけをみるとサンプリングにより高速化出来ている，と言える．しかし速度の他に本来対訳であるペアが誤って振り分けられないかという精度も重要である．次に処理速度と精度の関係について述べる．

3.4.3.2 サンプル数を変化させた時の計算コストと精度の変化

サンプル数を変化させた時の計算コストと精度の測定結果を示す．多重度 1 の場合の計算コストの変化を図 3.3 に，誤分類率の変化を図 3.4 に示す．また多重度 3 の場合の計算コストの変化を図 3.5 に，誤分類率の変化を図 3.6 に示す．

まず多重度 1 の場合を見ると，計算コストはサンプリングの方が小さく安定しているのに比べ，k-medoid 法はサンプル数が増加すると徐々に計算コストが増大してしまっている．これは k-medoid 法は medoid が収束するまで実行されるための計算コストであると思われる．サンプル数を大きくすればするほど，medoid の収束に時間がかかっているのが見て取れる．ただサンプリングも k-medoid 法もおよそ 70 %と非常に高い確率で誤分類を起こしており，多重度 1 の場合は確かに計算コストは低くなるが精度の面であまり使い物にならない，と言える．

次に多重度 3 の場合を見てみる．計算コストははじめは k-medoid 法の方が低いが，サンプル数が増えるに従い k-medoid 法の計算コストは徐々に増大している．一方サンプリングの場合は計算コストが下がっていることが見てとれる．つまり k-medoid 法は k の値が大きくなるに従い収束しづらくなり振り分けに時間がかかりすぎてしまう，と言うことが出来る．また誤分類率に関しても，全体的にサンプリングの方が低くなっているのが見て取れる．この最も大きな原因は，各クラスタの大きさである．k-medoid 法は各クラスタの大きさを全体的に均等にするが，そのために誤分類率が増えてしまっている．一方サンプリングの方が大きなクラスタが出来ていてその中に多くのテキストが入っているため誤分類率は低く抑えられている．大きなクラスタは全対全比較の時の計算コスト増大の要因となるものであるが，k-medoid 法はコストをかけて均等に振り分けても誤分類率が大きくなるだけであるという結果が得られた．つまり，計算コストと精度両方の面でサンプリングの方が良い成績となってしまっている．

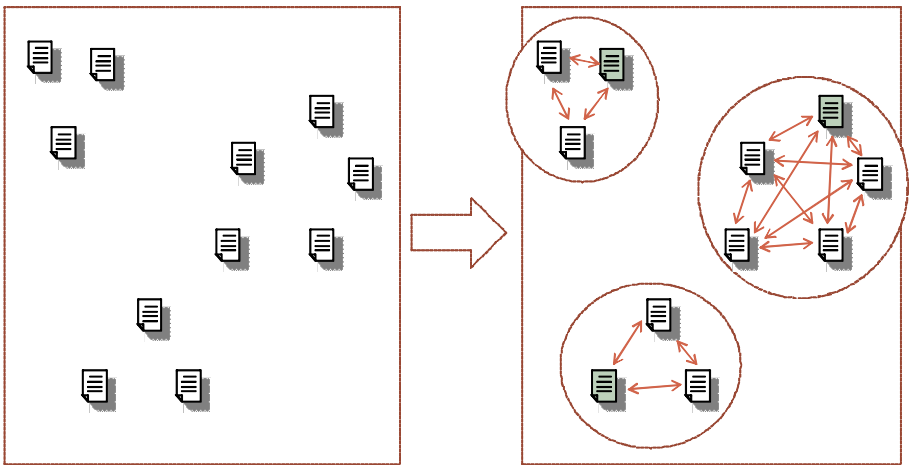

図 3.1: サンプリングの概要

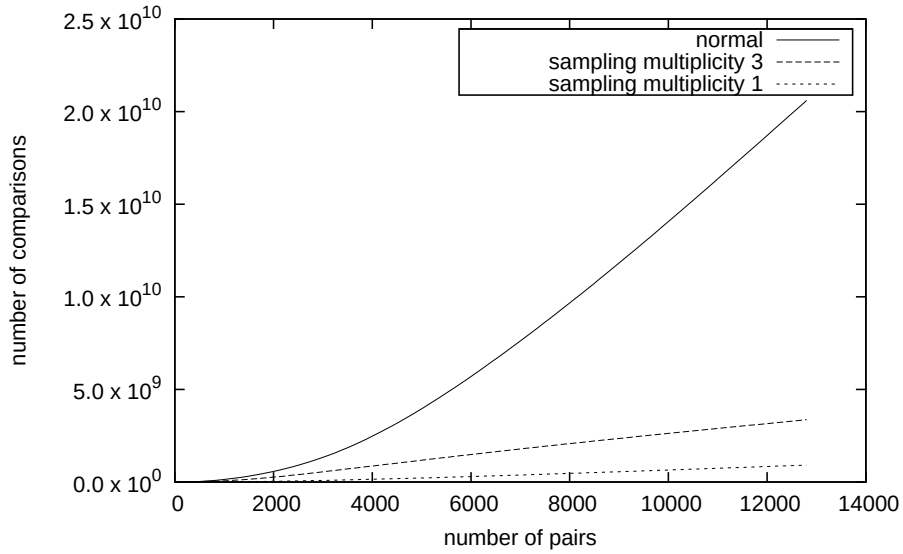

図 3.2: サンプリングによる処理速度

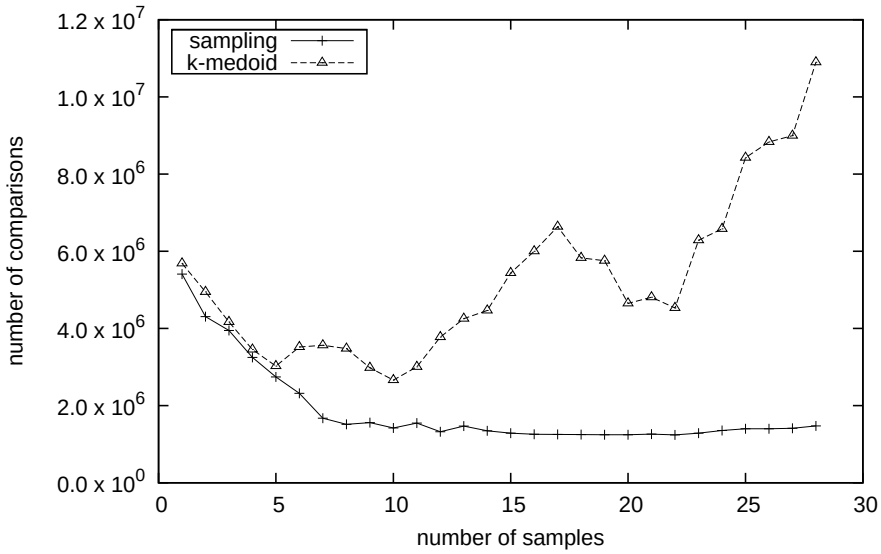

図 3.3: 多重度 1 の場合の処理速度

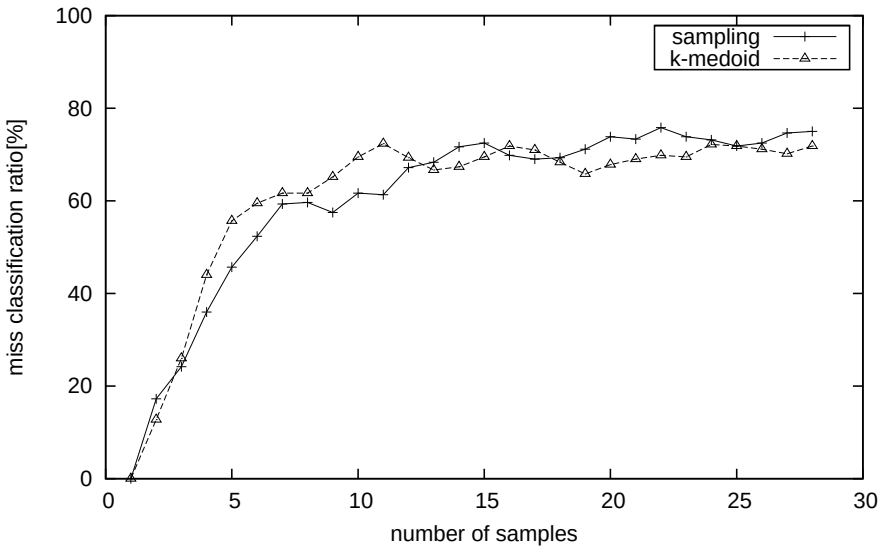

図 3.4: 多重度 1 の場合の誤分類率

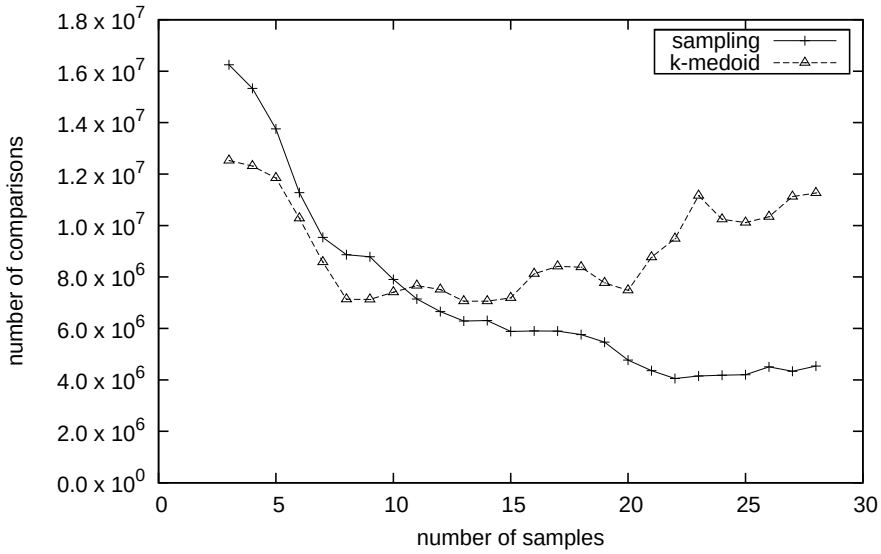

図 3.5: 多重度 3 の場合の処理速度

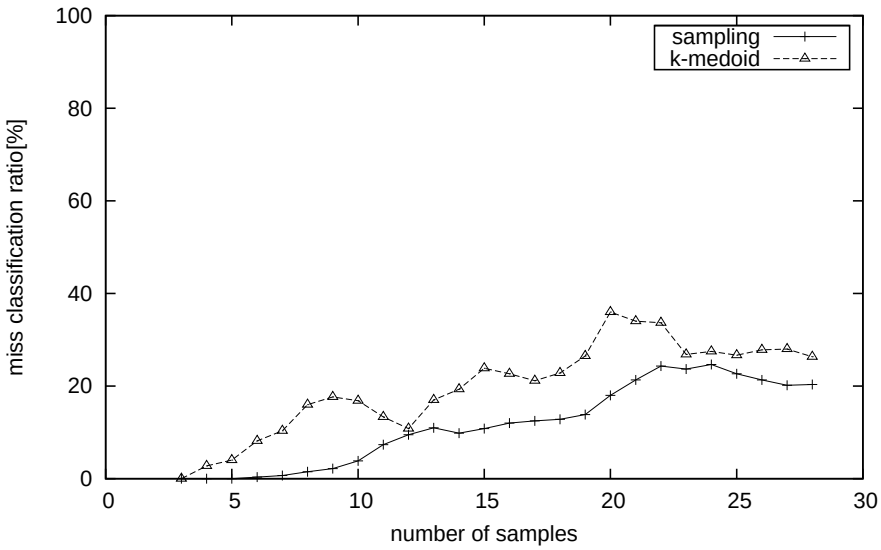

図 3.6: 多重度 3 の場合の誤分類率

3.4.3.3 多重度・計算コスト・誤分類率の関係

次に，多重度と計算コストの関係を図 3.7 に，多重度と誤分類率の関係を図 3.8 に示す．多重度を変化させていった場合でも，全体的に k-medoid 法の方が計算コストが高く，また誤分類率も高くなっていることが分かる．これによると先の実験でも示したように，多重度によらず k-medoid 法による分割の方がサンプリングよりも悪い結果となってしまっている．

次に図 3.9 に両手法における計算コストと誤分類率の関係をプロットしたものを，図 3.10 に振り分けを全てのテキストに対して行い，全対全比較は並列に行った場合のグラフを示す．縦線は全対全比較を行った場合の処理速度で，この縦線より左にある点が高速化が出来ていることになる．また，点が左下にあるほど処理速度・精度共に良いと言うことが出来る．サンプリングの方が安定して左下にプロットされているのが見て取れる．また，計算コストは大きく分けて振り分けのコストとクラスタ内全対全比較のコストの 2 つに分けられるが，並列化により全対全比較のコストが大きく削減されることが示されたので，今後振り分けの計算コストを削減することで全体的な高速化が図れると思われる．

以上より，サンプリングと k-medoid 法を比べた場合はサンプリングの方が良い結果が得られた．k-medoid 法を用いることによる利点は振り分け後のクラスタが均等に小さくなることが挙げられるが，これが振り分けにかかる計算コスト及び誤分類率による影響と比べて小さいことが k-medoid 法が成果が出なかった原因であると思われる．以降の実験では，サンプリングを用いる方向で話を進める．また，図 3.9 によると多重度やサンプル数といったパラメータで誤分類率及び計算コストの調整が出来，またトレードオフの関係にあることが示されたので，今後計算コストを削減することで相対的に誤分類率を抑えることが出来ると思われる．特に振り分け後の全対全比較の処理コストは並列化により高速化出来るので，以降並列化が困難な振り分けの計算コストの削減を目指す．

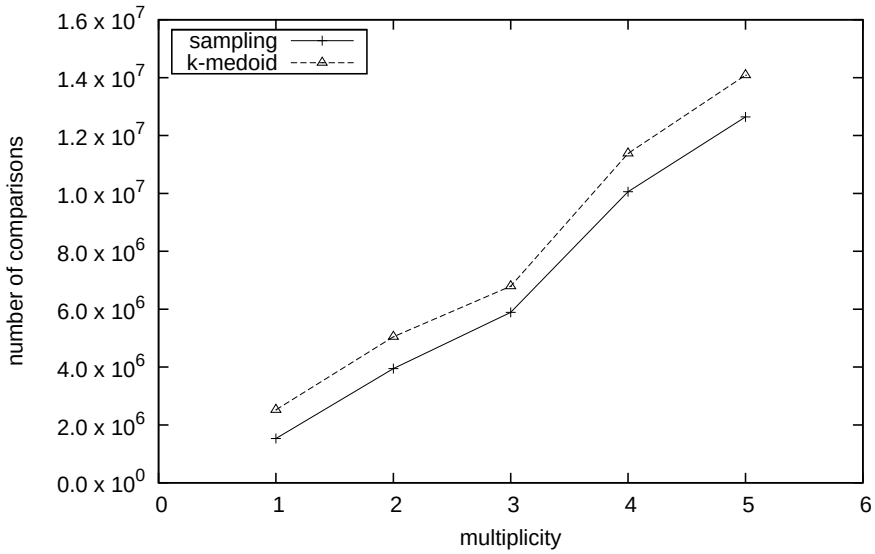

図 3.7: 多重度と処理速度の関係

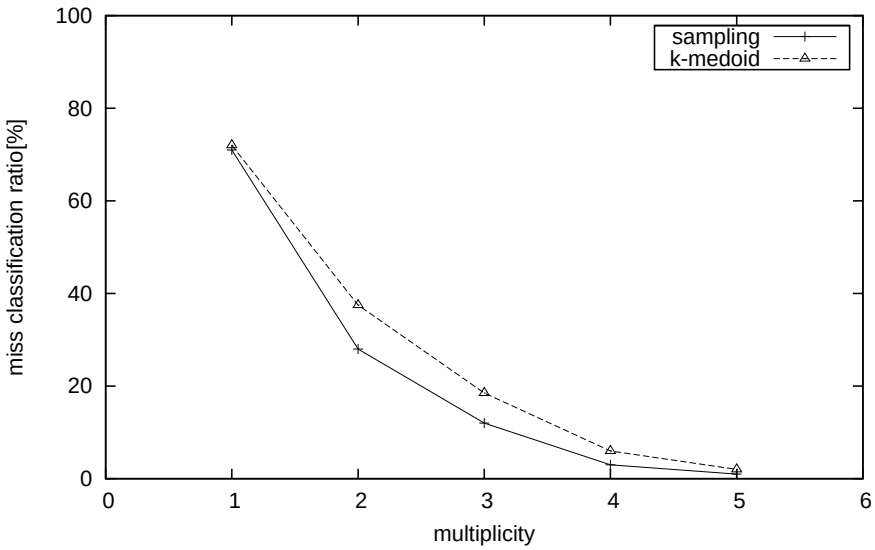

図 3.8: 多重度と誤分類率の関係

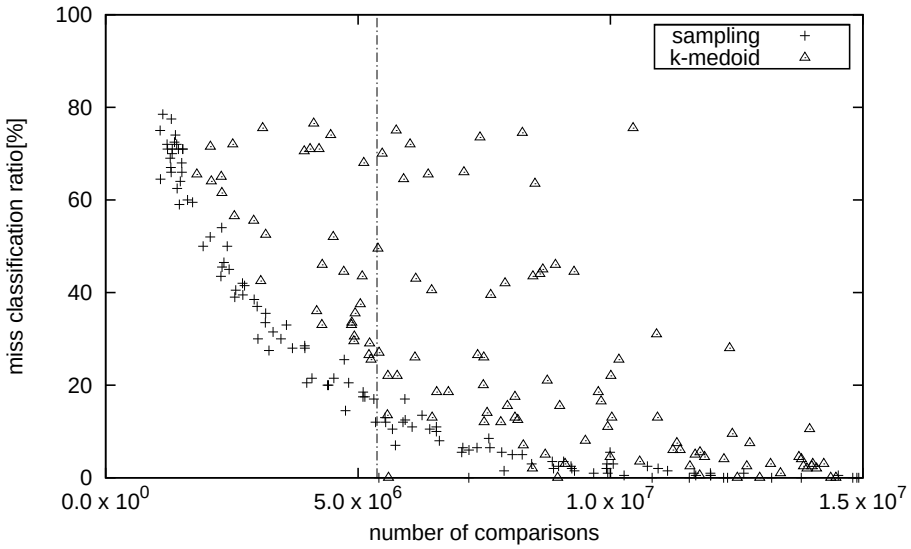

図 3.9: 処理速度と誤分類率の関係

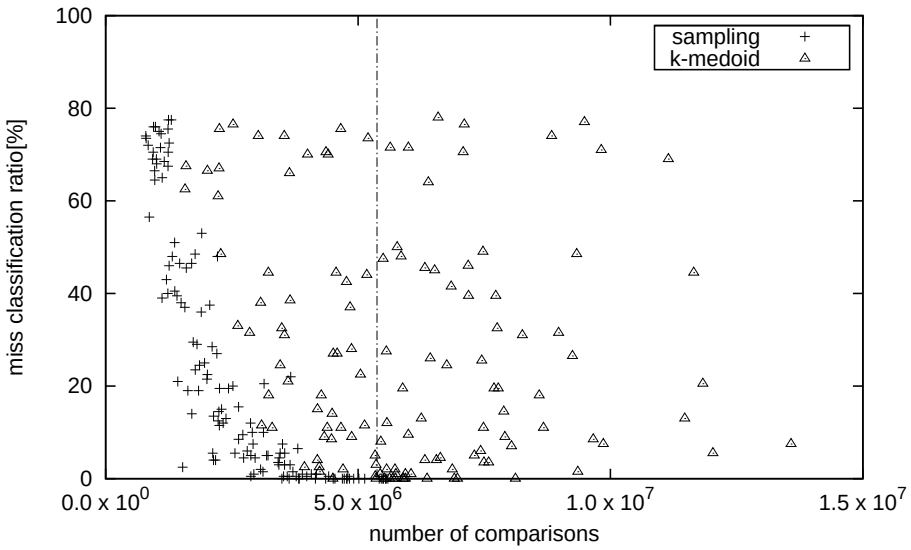

図 3.10: 並列化した時の処理速度と誤分類率の関係

第4章　LDAによる対訳テキスト抽出

本章では対訳テキスト抽出の高速化を目的とした手法として，LDA を用いた次元削減について説明し，評価を行う．

4.1　LDAによる次元削減

2.4 で対訳辞書のグラフ構造を用いることで，テキストをおよそ 9,000 次元の単語空間で表すことを示した．この単語空間中には対訳判定を行う上で特に重要な単語が存在すると考えられ，そのような単語を抽出した空間で対訳判定を行うことで更に計算コストが削減されることが期待される．ここでは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) 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．

4.1.1　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 2.3 で述べた α, β を推定するものである．特に今回扱うのは β，つまり ‘各トピックにおける単語の生成確率’ であり，トピック毎にそのトピックを特徴付ける語が確率の形で得られる．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，トピック毎に生成確率が高い単語を抽出する．

LDA による単語抽出を行う場合はパラメータとして

- トピック数
- 各トピックにおける生成確率が上位の単語の抽出数

の２つが考えられる．これらのパラメータを変化させることでアウトプットとなる単語抽出数の調整が可能である．

4.1.2　LDAを用いることによる利点

LDA はトピック分類のアルゴリズムであり，2.3 に示すような特徴を持つ．様々なアルゴリズムがある中で LDA を採用した理由は

- 複数トピックに対応
- 精度が高い
- アウトプットが扱いやすい

等が挙げられる．

‘複数トピックに対応’ とは，扱うテキストの制限が明確に定まっていない関係上，学習対象となるテキストが 1 トピック文書とは限らないことから，単独トピックを対象にしたアルゴリズムよりも複数トピックに対応したほうが精度が上がるだろう，という考えからである．

‘精度が高い’ とは，先の ‘複数トピックに対応’ とも関連するが，LDA は 2003 年と比較的最近に提案された手法であり，近年のテキスト分類の手法の中でも精度が高く出力が安定しているものであると言える．例えば [3] では，他のよく用いられるマルチトピック統計的言語モデルである PLSI [11] に比べ高い精度が得られた，としている．

‘アウトプットが扱いやすい’ とは，LDA の出力として各単語の各トピックにおける生成確率を得ることが出来るため，容易に単語間の比較による有用な単語の抽出が可能となる．

以上により，LDA を採用した．

4.2 実験

4.2.1 実験概要

実験データ及び実験マシンは 3.4.2 と同様である．

まず Fry の WebCorpus の日英対訳ページ 6400 ペアから後に評価に用いる 200 ペアのテキスト群に対して，トピック数と各トピックの上位単語数をパラメータとして単語抽出を行った．この時のトピック数と上位単語数の変化による単語数，及び各単語数で Fry の WebCorpus の 200 ペアに対して全対全比較をした時の計算コストを測定した．次に抽出された単語のみを用いて，Fry の WebCorpus の 200 ペアに対して多重度を 3 として単語比較回数の変化の測定，及び計算速度と精度の変化の測定を行った．

4.2.2 実験結果

4.2.2.1 LDA により抽出される単語の評価

Fry の WebCorpus6400 ペアに対してパラメータを変動させて単語抽出を行った時の単語数の変化を，図 4.1 に示す．各グラフは与えたトピック数のみが異なり，横軸は各トピックの上位単語をどれだけとるかである．また，縦軸は出力された単語の内重なりを除いたものである．グラフを見ると，トピック数の増加・各トピックの上位単語数の増加に伴いほぼ線形に増加していることが分かる．これにより，例えば出力単語を 300 語としたい場合はトピック数 10 の Top80 やトピック数 15 の Top50 を選べば良い，といったように取得したい単語数に応じたパラメータ設定が出来るようになる．

また，各単語数に対して全対全比較を行った時の単語の比較回数を図 4.2 に示す．単語数を増やせば単語比較回数が増えるのは当然であるが，単語数の増加に対する単語比較回数の増加率が 200 語〜300 語辺りから徐々に収束し始めているのが見て取れる．これはテキスト中に頻出する重要単語が優先して選択されていることを示していると考えられる．つまり，一定数以降は単語選択数を増やしても処理速度への影響が小さくなっている，と言える．

次に，トピック分類による単語出力の例を表 4.1 に示す．この例はトピック数 20 の Top10 を選んだ時のあるトピックにおける単語である．本来 1 行 1 単語であるが，意味 ID →単語の変換を行っているので，1 行に日英様々な単語が含まれているのがわかる．この例をみると研究・医師・細胞・遺伝子等の単語が含まれていることから医学・生物学系の記事のトピックではないか，と見ることが出来る．

今回はこの実験結果から，表 4.2 のような 4 種類の次元削減＋次元削減しない場合，の合計 5 つの場合についての実験を行う．

4.2.2.2 LDA で抽出された単語を用いた全対全比較の計算コスト

まずは全対全での対訳判定を行った時の計算時間の変化の測定結果を表 4.3 に示す．ここで示される数値は 2 値比較の回数で，この数値が低いほど処理は早く終わることになる．

この表が示すように，対訳判定における単語の比較回数は次元を削減すればするほど減少していき，100 単語まで次元削減を行った空間では何も次元削減を行わない場合と比べて比較回数がおよそ 15 %にまで削減されていることが分かる．以上より，次元削減を行うことにより計算量が削減される，と言える．

4.2.2.3 LDA で抽出された単語を用いたクラスタリングの計算コストと誤分類率

次に，次元削減による計算量の変化を図 4.4 に，誤分類率の変化を図 4.3 に示す．ここで横軸はサンプル数である．これら 2 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．

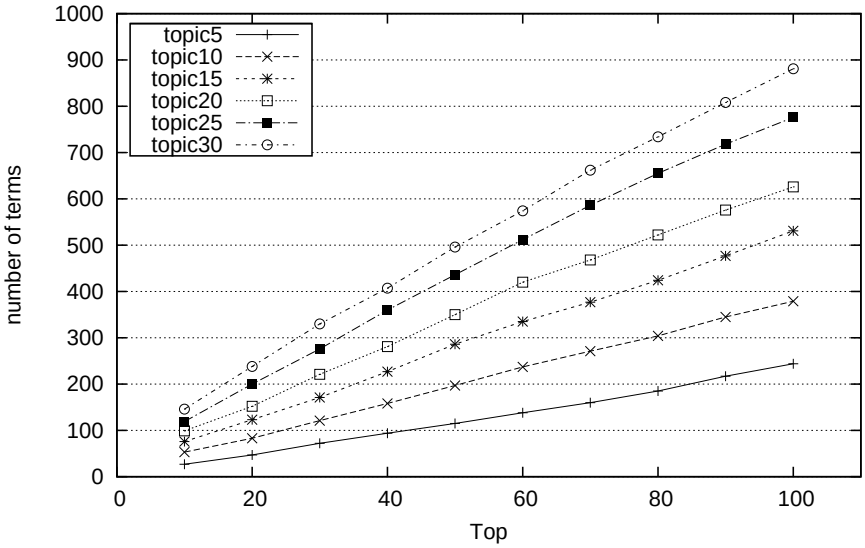

図 4.1: LDA で抽出された単語数

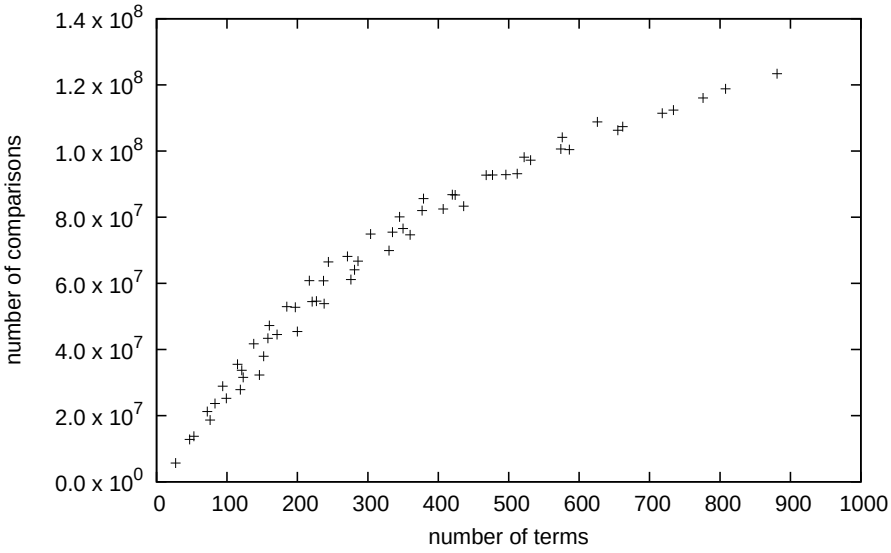

図 4.2: LDA で抽出された単語数と全対全比較の処理速度の関係

id	terms
03005	den, research, study, アジト, エチュード, 研究, 習作, 書斎, 練習曲
00476	doctor, internist, physician, 医師, 内科医, 博士
09397	ability, accommodation, administration, advantage, army, authority, benefit, capability, child, children, competence, consequence, convenience, corruption, denomination, descendant, effect, efficacy, facility, faculty, force, forces, fruit, function, gain, gains, graft, influence, inquiry, intensity, interrogation, issue, kid, might, momentum, offspring, outcome, posterity, power, problem, product, profit, progeny, query, question, quiz, reference, result, return, scion, skill, strength, talent, tenor, troops, unit, validity, workings, アドバンテージ, タレント, テナー, テノール, フルーツ, 影響, 影響力, 汚職, 果実, 果物, 課題, 関数, 機能, 儀式, 技量, 疑い, 疑問, 強さ, 強み, 教派, 軍, 軍勢, 軍隊, 結果, 権威, 権限, 権力, 後裔, 効きめ, 効き目, 効果, 効能, 効力, 好都合, 行政, 才能, 作用, 参考, 産物, 仕事率, 子, 子供, 子孫, 施政, 児童, 質問, 実, 取り調べ, 手当, 手腕, 趣旨, 収益, 収穫, 出典, 所産, 乗積, 職能, 職務, 人材, 勢い, 勢力, 成果, 成績, 政権, 政治, 製品, 接ぎ穂, 妥当性, 大家, 単位, 典拠, 転訛, 動力, 得, 内閣, 背徳, 部隊, 返り, 便宜, 便利, 末裔, 儲け, 木の実, 問い, 問題, 利益, 利点, 利得, 力, 力量, 冪
09409	aim, concept, conception, foci, focus, idea, intent, intention, mark, notion, object, objective, purpose, target, thinking, thought, もの, マーク, マルク, 意向, 意思, 意図, 概念, 観念, 記号, 計画, 考え, 志, 思いつき, 思考, 思想, 出発点, 焦点, 跡, 対象, 着想, 的, 点数, 標的, 物, 物体, 目印, 目的, 目的語, 目標
09331	he, helium, humankind, husband, man, mankind, masculine, men, person, ヘリウム, 身体, 人, 人間, 人称, 人類, 男, 男性, 夫
09392	affair, auto, box, car, case, conduit, container, declension, duct, happening, incident, pipe, receptacle, tube, vessel, watercraft, wheel, キセル, ケース, コンセント, コンテナー, ダクト, チューブ, ハプニング, パイプ, 格, 活字ケース, 患者, 管, 器, 事件, 事例, 自動車, 車, 車両, 車輪, 出来事, 鞘, 乗用車, 場合, 船, 船舶, 笛, 筒, 導管, 入れもの, 入れ物, 箱, 物入れ, 問題, 容器, 例
09389	airframe, annotation, arrow, axle, belly, body, bole, build, comment, commentary, composition, constitution, construction, explanation, explication, figure, form, formation, forme, grip, haft, handle, handlebar, hilt, interpretation, make, mode, nose, organization, physique, prow, shaft, shape, snout, stem, stomach, structure, style, system, trunk, こく, グリップ, シャフト, シュテム, スタイル, トランク, ハンドル, ファッション, フィギュア, ボディー, 握力, 胃, 花柱, 解釈, 解説, 幹, 幹線, 機構, 機体, 形, 形式, 形状, 形相, 形態, 系, 系統, 建造物, 構成, 構造, 構造物, 作文, 作法, 姿, 姿態, 軸, 車軸, 取っ手, 樹幹, 書式, 心棒, 身体, 人影, 人物, 図, 図形, 数字, 雛形, 旋法, 船首, 船首材, 組み版, 組み立て, 組織, 組立て, 体, 体格, 体系, 体制, 団体, 注解, 注釈, 筒先, 胴, 胴体, 道具方, 肉体, 鼻, 評言, 付注, 符尾, 腹, 腹部, 物体, 文体, 柄, 編隊, 方式, 方法, 矢, 矢印, 容姿, 様式, 用紙, 葉柄, 流儀, 論評, 舳先
02698	gender, sex, セックス, 性, 性別
01475	cell, guardianship, sickroom, ward, wardship, 監房, 後見, 細胞, 電池, 独居房, 被後見人, 病室
05344	gene, ゲン, 遺伝子

表 4.1: LDA により抽出される単語の例

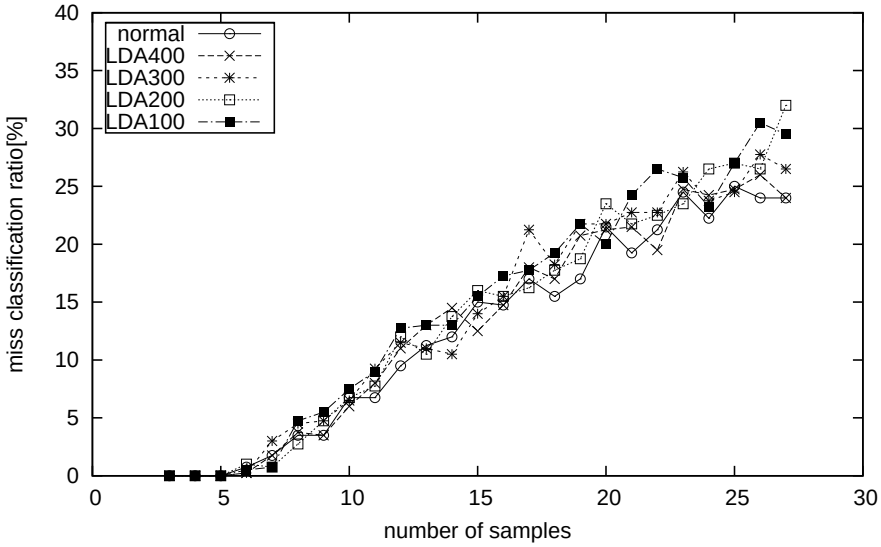

図 4.3: LDA で抽出された単語を用いた時の誤分類率

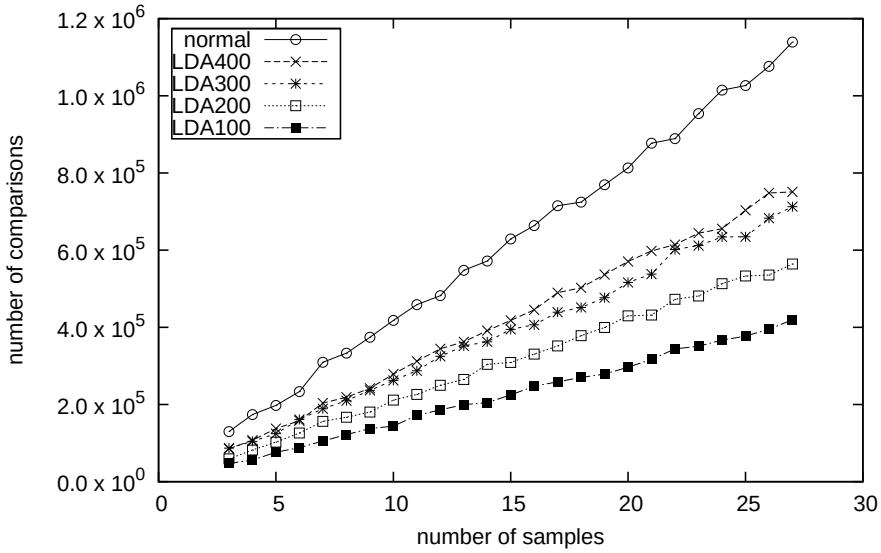

図 4.4: LDA で抽出された単語を用いた時の処理速度

name	# of terms	topic	top
lda100	99	20	10
lda200	200	25	20
lda300	304	10	80
lda400	407	30	40

表 4.2: LDA の評価のために用いる単語数とパラメータ

type	normal	lda400	lda300	lda200	lda100
# of comp	1.69×10^8	8.24×10^7	7.49×10^7	4.54×10^7	2.52×10^7

表 4.3: LDA で抽出された単語を用いた時の全対全比較における処理速度

第5章　転置インデクスによる対訳テキスト抽出

3章及び4章では大量に存在するテキストの中から対訳ペアを抽出する手法について説明した．しかし対訳コーパスから対訳テキストを抽出するといった場合，大量のテキストから多数の対訳ペアを集める要求の他に，任意のテキストを検索元としてそのテキストの対訳が既に存在するテキスト群から検索するという要求も考えられる．そこで本章では，１つのテキストをクエリとしてそのテキストの対訳を求める手法について説明する．その後，対訳テキスト検索を応用した対訳テキスト抽出の高速化を述べる．

5.1　対訳テキスト検索

5.1.1　転置インデクス

転置インデクス [23] とは，Google[1]のようなキーワード型検索エンジン構築によく用いられる手法で，単語→テキスト の対応付けを行いインデクス化したものである．例えばテキスト A には単語 1,2 が，テキスト B には単語 1 が，テキスト C には単語 1,2,3 が含まれていた場合，転置インデクスは 1 → (A,B,C), 2 → (A,C), 3 → (C), となる．転置インデクスを用いることで検索に与えられたキーワードから高速にキーワードを含むテキストを抽出することが出来る．

通常は検索エンジンなど大規模なテキストを保持している場合でないと単語の出現割合の関係上転置インデクスが非常に疎なものとなってしまったり，同じ意味の単語であっても単語自体は異なるため別の扱いとなってしまうなどの問題が生じるが，提案手法の場合は各テキストの単語を意味 ID に変換して同意語をまとめているので，意味的に同じ単語に対して高速な検索を行うことが出来る．

5.1.2　転置インデクスを用いた対訳テキスト検索

転置インデクスを用いた対訳テキスト検索について説明する．状況として，対訳を発見したい特定の言語で記述されたテキストがあり，データとして様々な言語で記述されたテキストが既に存在しているとする．またこのテキストは意味 ID への変換及び意味 ID の転置インデクスが生成されており，また各テキストの意味 ID 列の長さも別データとして保持しておく．

基本的な流れは

[1] http://www.google.com

- クエリの意味 ID 変換
- 意味 ID から転置インデクス情報取得
- ランク付け

となる．それぞれについて説明していく．

5.1.2.1 クエリの意味 ID 変換

まずはクエリを意味 ID に変換する必要がある．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す．具体的な手順は 2.4.2 に書いてある通りで，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．

データの流れは

$$\text{(入力) クエリテキスト} \rightarrow \text{(出力) 意味 } ID \text{ 列}$$

となる．

5.1.2.2 意味 ID から転置インデクス情報取得

意味 ID の数列が与えられたら，次は転置インデクス情報を取得する．転置インデクスは事前に構成されていて

$$\text{意味 } ID : ((\text{出現数}, \text{テキスト } ID), (\text{出現数}, \text{テキスト } ID), ...)$$

のデータ構造を持っている．これは，各意味 ID が各テキストにどれだけ出現するかを意味 ID を key にして検索できる，という意味である．意味 ID 列から転置インデクス情報を取得することで，‘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．

データの流れは

$$\text{(入力) 意味 } ID \text{ 列} \rightarrow \text{(出力) 転置インデクス情報のリスト}$$

となる．

5.1.2.3 ランク付け

転置インデクス情報のリストを各テキスト毎にまとめ直し，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．概要を図 5.1 に示す．ここでランク付けに用いるスコアの要件として

- 対訳の可能性が高いほどスコアが高くなる
- 転置インデクス情報から計算することが出来る

の２点が挙げられる．ここで１番目の要件は trans_score を用いることで実現可能であると思われる．２番目の要件も，trans_score は基本的に両テキスト間の一致する単語と両テキストの長さの比で表すことが出来るので，単語間の距離を考えないことを差し引くと trans_score で実現可能である．

テキスト間の類似度を表す式は以下になる．

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

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テキスト間の距離の指標とし，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 と近い距離尺度を用いて計算することができる．またこの inv_score でソートすることで距離が近いテキストが抽出できる．またクエリテキストと残りのテキストについて，クエリに出現する単語と一致するもののみをカウントするので，従来の trans_score よりも計算コストが削減されることが期待される．

データの流れは

(入力) 転置インデクス情報のリスト→ (出力)inv_score

となる．以上により，クエリテキストと各テキストとの距離計算が完了する．

5.2 対訳テキスト抽出の高速化

本節では，5.1 で説明した対訳テキスト検索を応用し 3 章及び 4 章の手法と組み合わせた手法について説明する．

5.2.1 転置インデクスを用いたクラスタリングの高速化

5.1 節で転置インデクスを用いることで対訳テキスト検索が高速化されることを示した．これを対訳テキスト抽出に応用する．

転置インデクスを用いた検索で ‘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．ただし，ここでのランク付けに用いるスコアは trans_score から単語間距離情報を除去した inv_score である．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クラスタに振り分ける段階とクラスタ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距離尺度を用いる必要がないため inv_score を用いて高速化する．

具体的な流れを以下に示す．

1. サンプルテキストをランダムに選択
2. サンプルテキストをクエリとして全テキストに対して検索・スコア付け
3. 最も近いサンプルテキストのクラスタに振り分け
4. クラスタ内で全対全の対訳判定

サンプリングとは，2 が ‘全テキストとサンプルテキストとの距離を計算’ であることが異なる．つまり転置インデクスを用いることで全テキスト数×サンプルテキスト数の trans_score 計算を検索処理に置き換えるわけである．これにより距離の定義が ‘trans_score’ から ‘inv_score’ にはなるが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．

これにより高速化出来る理由として，単語比較回数に注目する．代表点となるテキストを q_i, 探索するテキストを t_j, クラスタ数を c, テキスト数を n とすると，通常の trans_score の場合は，テキストを T1・T2，それぞれの単語数を len(T1)・len(T2) とした時に

$$\sum_{i=0}^{c}\sum_{j=0}^{n} len(q_i) + len(t_j)$$

だけの単語比較を行う必要がある．これを転置インデクスを用いた検索処理に置き換えた場合，クエリに含まれる単語数だけ転置インデクスからデータを取得することとなり，結果として ‘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る．つまり

$$\sum_{i=0}^{c}\sum_{j=0}^{n} len(q_i \cap t_j)$$

だけの単語比較で済むことになる．

5.3 実験

5.3.1 実験概要

実験には Fry の Web コーパスで公開された URL のうち，アクセスして取得出来た全 13105 ペア，及びその中の 200 ペアを用いた．実験に使用したマシンは 3.4.2 と同様である．

まず inv_score の評価として，inv_score でランク付けを行った場合に対訳が上位に来るか，の評価を行う．具体的には Fry のコーパス 13105 ペアを日本語と英語に分けて，それぞれの日本語テキストに対して，英語テキスト 13105 全てと inv_score を計算し実際の英語の対訳が何番目に位置するのか，を測定した．比較として trans_score に対しても同等のことを行った．次に inv_score を計算する場合の計算コストについて，通常の trans_score を用いて全対全比較を計算する形で対訳テキストを探索する場合と比べてコストがどのように変化するか，の評価を行う．

次に，転置インデクスを用いた場合と通常のサンプリングの場合で，サンプル数を 14 で固定した時の多重度・計算コスト・誤分類率の関係を測定した．具体的な方法は 3.4.3.3 と同様で，転置インデクスを用いた場合とサンプリングで評価を行う．また特に振り分けにかかる計算コストについての評価を行う．

5.3.2 実験結果

5.3.2.1 inv_score の評価

inv_score で対訳テキストのランク付けを行った．結果を表 5.1 に示す．結果を見ると，inv_score で実際の対訳が Top1 となるのは全体の 3 割とあまり高くなく，8 割〜9 割のカバー率となるのは Top100〜1000 テキストを見た場合である．Top1000 テキストを見ればほぼ正解が表れると見てよいと思われる．また trans_score で Top1 に実際の対訳が現れるのはおよそ 6 割となっており，やはり単語間の距離情報を削減している分 trans_score の方が良い性能が出ている．このように多少精度は下がってしまうが，計算速度向上のために振り分けには inv_score を用いて，振り分け後のクラスタ内で対訳候補を抽出する際には trans_score を用いる．

次は計算コストの評価を行う．ここでは 200 テキストに対して検索を行う場合を考える．クエリとして与えられた時にテキストの対訳を発見する場合，通常はテキスト間で全対全で trans_score 計算を行い出力をソートする．また提案手法の場合はクエリに含まれる単語を転置インデクスから取得し，inv_score を計算してソートする．

全対全比較の場合は約 2,600,000 回の意味 ID の 2 値比較が必要になったのに対し，提案手法の場合はランク付けに約 140,000 回，Top1000 のテキストに対し詳細な意味 ID 検索を行った場合で約 200,000 回の意味 ID の 2 値比較となったので，合計しても約 340,000 回，およそ 10 倍程度高速化出来ていることになる．

5.3.2.2 対訳テキスト抽出の高速化の評価

多重度と計算コストの関係を図 5.2 に，多重度と誤分類率の関係を図 5.3 に示す．これらを見ると，誤分類率にはほとんど影響を与えることなく全体的に転置インデクスを用いている方が計算コストが削減されているのが分かる．計算全体としてはこの計算コストの利点はあまり大きくないかもしれないが，ここで削減されているのは振り分けの計算コストであり，振り分け後は全体的に並列化が可能であることから，振り分け部分の高速化は特に重要であると言える．

それを示したのが次である．転置インデクスとサンプリングについての，特に振り分けにかかる計算コストの結果を図 5.4 に示す．この時の多重度は 3 である．振り分けにかかるコストが，サンプリングに比べても 5 倍以上高速になっていることが見てとれる．このように転置インデクスを用いることで振り分け処理を高速化することが可能となる．

最後に，サンプリングと転置インデクス，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 5.5 に，並列化した時の計算コストと誤分類率の関係を図 5.6 に示す．通常のサンプリングに比べて転置インデクスを用いた方が，また転置インデクスに比べて LDA も用いた方が高速化されているのが見てとれる．例えば図 5.5 では，誤分類率を 5 %程度に抑えた場合で処理速度が全対全比較と比べ 2 倍程度速くなっている．また転置インデクス及び転置インデクス＋ LDA のグラフが全体的に左下によっているのが分かる．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るということで，性能が良くなっている，と言うことが出来る．また並列化を施すことで転置インデクス＋ LDA を用いた場合は，全対全比較と比べて１０倍以上高速化出来た．

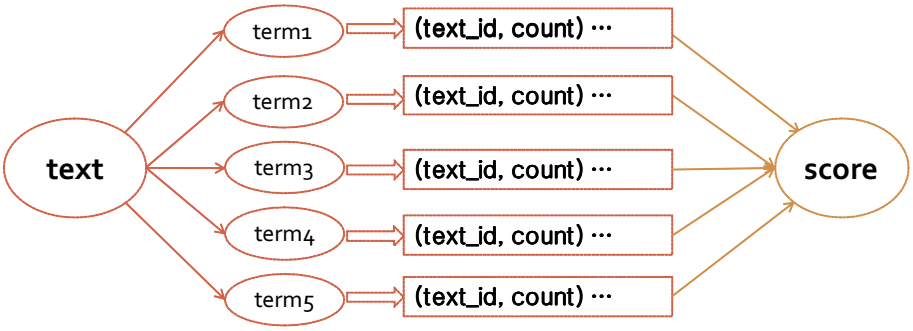

図 5.1: Inv_score を用いた対訳テキスト検索の概要

Top(/13105)	1	10	100	1000	10000
inv_score [%]	30.3	58.6	83.3	97.0	99.9
trans_score [%]	59.2	72.8	84.1	92.8	98.6

表 5.1: Inv_score と Trans_score の対訳カバー率

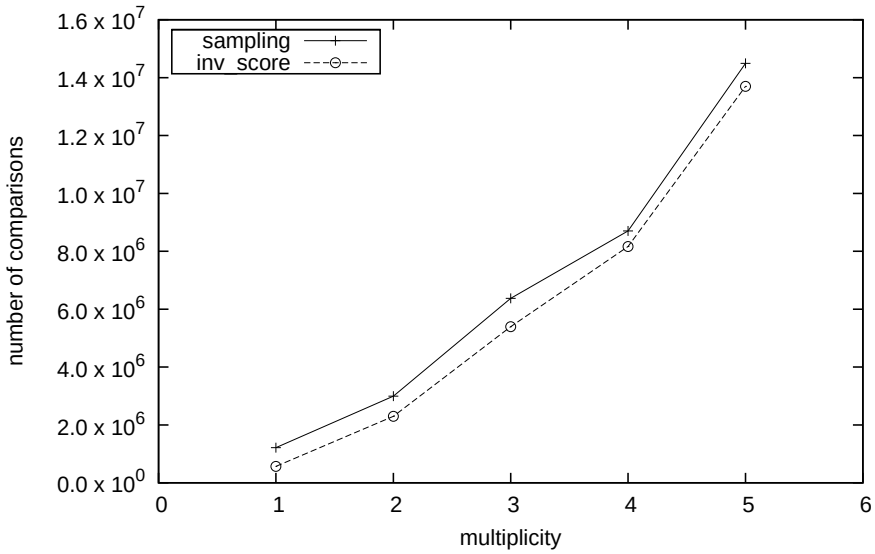

図 5.2: Inv_score の多重度と処理速度の関係

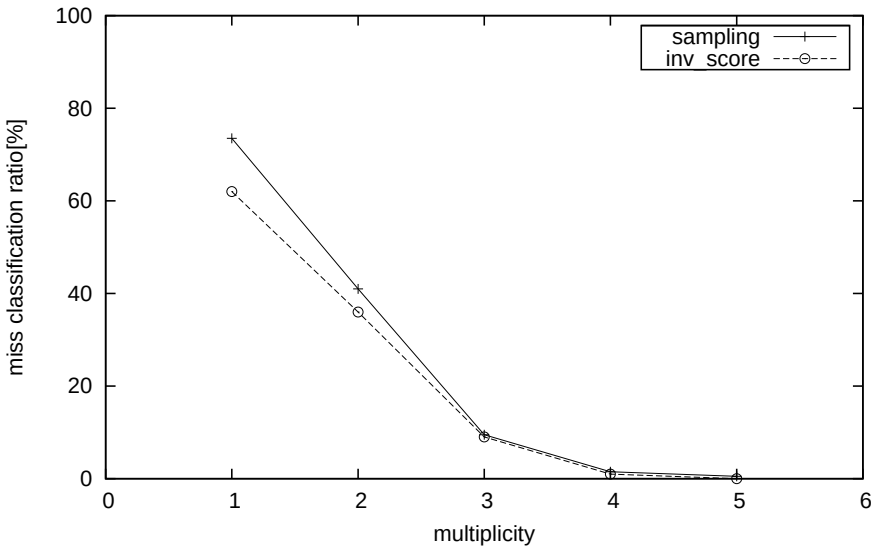

図 5.3: Inv_score の多重度と誤分類率の関係

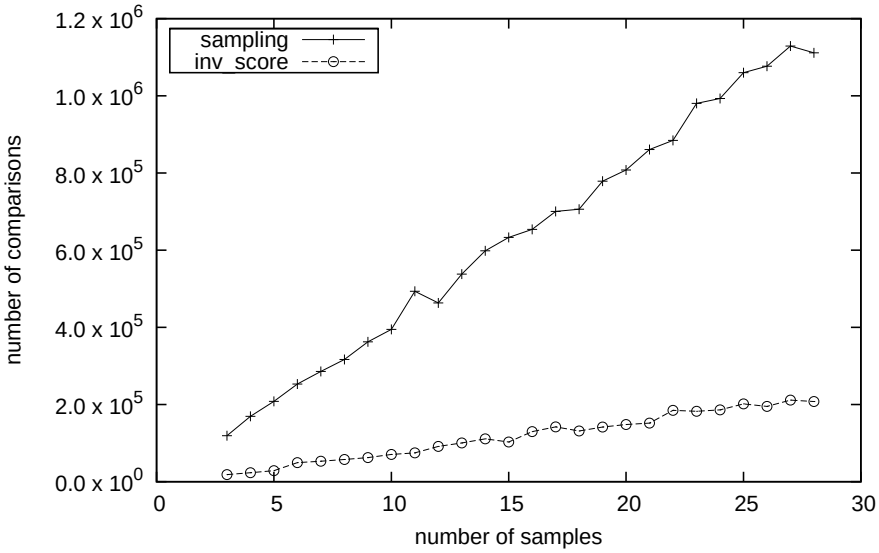

図 5.4: Inv_score による振り分けの処理速度

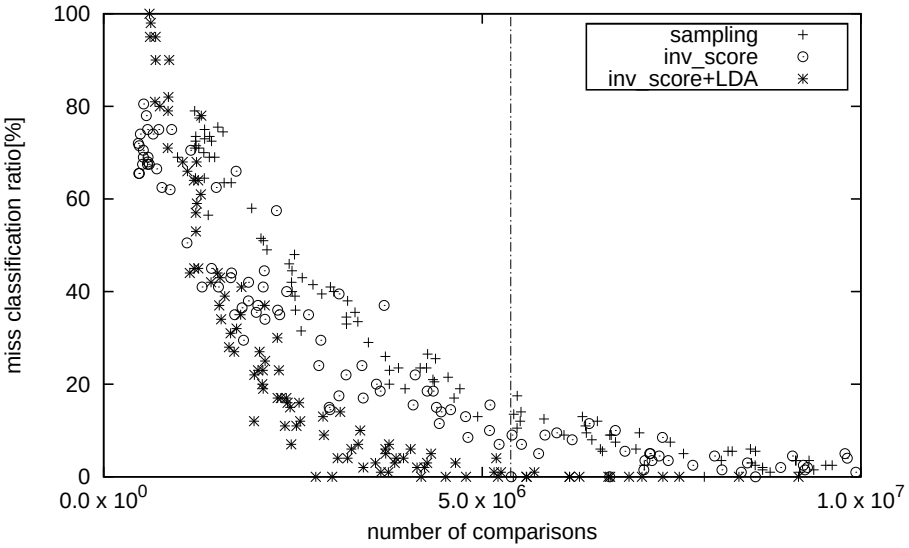

図 5.5: Inv_score 及び Inv_score+LDA の処理速度と誤分類率の関係

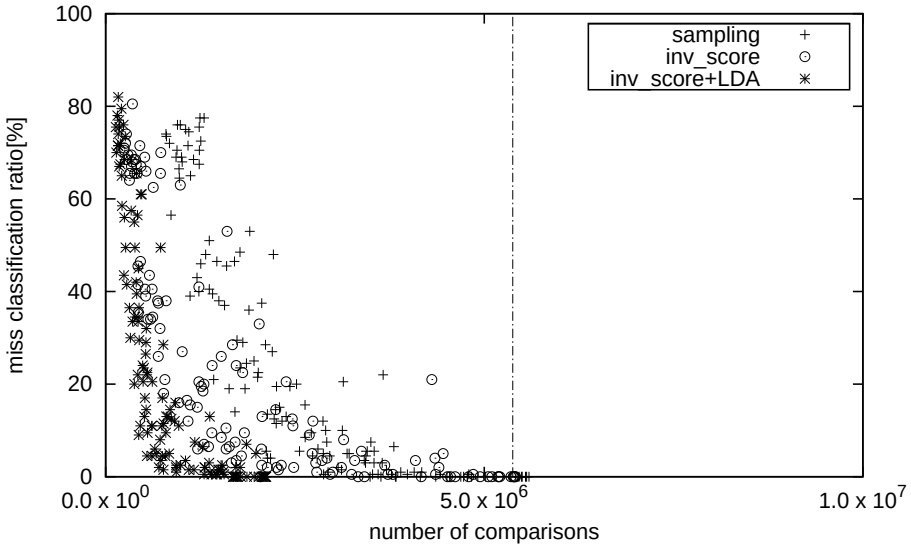

図 5.6: 並列化した時の Inv_score 及び Inv_score+LDA の処理速度と誤分類率の関係

第6章　大規模データへの適用

本論文で提案している手法は，大規模テキストコーパスを対象とした高速な対訳テキスト抽出を目指したものである．これまでにサンプリング・LDA による次元削減・転置インデクスによる高速化の提案と評価を行ってきたが，手法の精度を測定する時はデータとして 200〜400 程度の日英対訳ペアを用いており，実際に大規模なテキスト群からの対訳ペアの発見に関しては評価が十分出来ているとは言い難い．しかし最終的な目標を考えると大規模な状況に提案手法を適用することは重要な課題であると考えられる．そこで本章では大規模なテキスト群の中にいくつかの対訳ペアが存在する場合を想定して実験を行い，提案手法の大規模データにおける有効性を確認する．

6.1　大規模化により起こりうる問題

大規模なテキストを対象とする場合，小さなテキスト集合ではあまり考慮する必要が無かった問題が生じる可能性がある．その中でも特にメモリに関する問題について考える必要がある．そこでまず大規模化におけるメモリ問題について述べた後，提案手法の動作についての考察と評価を行う．

6.1.1　大規模化におけるメモリ問題

最も大きな問題はデータ量の増加に伴うメモリの問題である．つまり，テキストデータを変換した意味 ID 列を他のテキストと比較するためにはメモリに領域を確保しておかねばならないが，テキスト数が増えると全てのデータがメモリに載り切らなくなる，という問題である．この問題に対処するためにはテキスト集合を小さく分割する，あるいは各テキストに対する処理をまとめ，その処理が終わったらメモリを解放するといった処置が必要となる．全対全比較の場合は全てのテキストの意味 ID 列をメモリ上に載せる必要があり，テキスト数増加の影響が大きい．しかし我々の提案する手法の場合この問題を緩和出来る可能性がある．

まずサンプリングにおいて，サンプルテキストをランダムに選択してそれらに関しては意味 ID 列をメモリ上に載せる必要があるが，その後各テキストとサンプルテキストとの距離を計算しテキストを振り分けている段階では逐次処理が可能であり振り分けるクラスタを意味する (クラスタ ID, テキスト ID) のペアさえ保持していれば通常のテキストの意味 ID 列は不要となる．振り分け後は全対全比較が必要となるために各クラスタの全テキストをメモリに載せる必要があるが，これはクラスタ数を変更することで計算機のメモリに応じたテキスト数の調節が可能となっている．また LDA による次元削減を行うことで各テキストを表現する意味 ID 列が短縮され，結果としてより多くのテキ

ストを保持することが出来るようになる．しかし，転置インデクスを用いた手法は高速化のために各テキストの情報を転置インデクスの形で持たなければならずメモリ量は大きくなってしまう．LDA によりこの影響は緩和可能ではあるが，より大規模化に適用するためには転置インデクスの持ち方を工夫する必要があると思われる．

6.2 実験

6.2.1 大規模データ

今回の実験で使用する大規模データについて説明する．本研究は大規模な Web データを対象とするため，今回実験で使用するデータも実際に Web から持ってきたデータを利用する．具体的には 2.1.1 で用いた URL リストの元となったデータである．つまり我々の研究室で開発している Shim-Crawler を用いて Open Directory Project の大学関連ニュースページを seed としてクローリングを行ったページ群であり，大規模データの扱いのためその中からランダムに 100,000 ページ抽出した．seed ページこそ大学関連のニュース記事と内容に偏りがあるように見られるが，ページ収集→リンク抽出を 12 回程度繰り返し行っているため，およそ 9000 万ページのテキスト集合全体としてはほぼ偏りの無く Web 上に存在する様々な種類の Web ページを収集出来ていると仮定して良いと思われる．

6.2.2 実験概要

発見すべき対訳として Fry のコーパス 800 ペアを 100,000 ページのテキスト群に追加し，合計 101,600 ページの大規模テキストデータに対して提案手法を適用することにした．理想は大規模テキスト群から対訳テキストを抽出することだが，得られた結果の，特に精度に対する評価が困難であるために今回はこのようなデータ設定とした．

このテキスト集合に対して提案しているサンプリング・LDA・転置インデクスを組み合わせた手法を適用しクラスタ数を 20,40,..., 160 と 8 パターン，多重度を 2,4,...,16 と 8 パターンの計 64 パターンに対して処理速度と精度の関係を測定する．ここでの精度は実際の対訳が同じクラスタに分類される数である．

また，実装上問題となりうるメモリ使用量に関する考察を述べる．

6.2.3 実験結果

6.2.3.1 大規模データにおける処理速度と精度の関係

100,000 テキストから提案手法を用いて対訳テキストを抽出した時の実行時間と誤分類率の関係を図 6.1 に示す．プロットされた各点はそれぞれクラスタ数と多重度が異なる．また縦線が全対全で対訳テキストを比較した場合の実行時間である．

グラフを見ると，誤分類はほぼ存在せず実行時間が全対全で実行した場合に比べ半分以下になっている点がいくつか見て取れる．最も良い精度を示す点ではクラスタ数 40・多重度 4 で，誤分類が無く実行時間が約 2.8 倍早くなっている．これは大規模データの方がこれまで実験していたような小規模なデータに比べサンプリングによるオーダ n^2 の全対全比較の計算量削減の効果が大きいためであると思われる．またこれは並列化を施さない状態での結果であることから，並列化を施すことで更に高速化が図れることが期待される．以上より，大規模データに対しても提案手法が有効であることが示された．

6.2.3.2 メモリ使用量に関する考察

本手法を C++上で実装し大規模データに対して適用した場合，101,600 のテキスト群に対しては 320MB 程度メモリを消費した．また別の実験で 50,800 のテキスト群に対して同様の処理を行ったところメモリ消費量は 175MB 程度となった．このことから，テキスト数とメモリ使用量の比率がこれらの測定結果の比率に従うとするとメモリ 2 GB の環境ではおよそ 700,000 ページ程度のテキスト群に関しては特に実装上の工夫を行わなくても提案手法が適用可能であると思われる．

またそれ以上のテキスト群を扱う場合には，メモリ上には基本的にクラスタ関係の情報を載せて，テキスト自体は読み込んで振り分けが終わったらメモリ上から追い出すようにすることで逐次の対訳テキスト抽出が可能となる．ただしどのテキストがどのクラスタに属するのか，という関係性は保持する必要があるので，ファイルに書き出す等の処理が必要となる．また転置インデクスに関しては頻繁にアクセスされるものとほぼアクセスされないものがあるはずなので，LRU(Least Recently Used) アルゴリズム等を用いて明示的にアクセスされないものを間引く処理を行うことで効率化が図れるものと思われる．

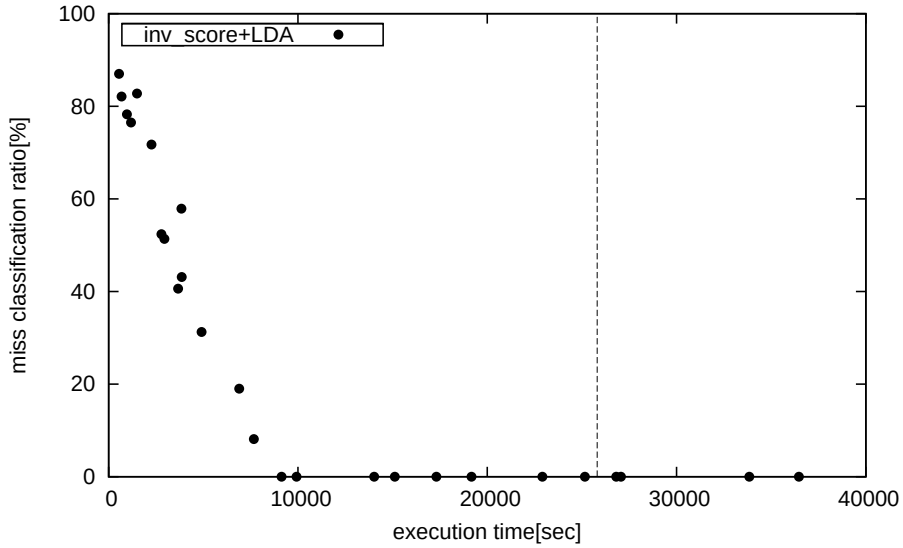

図 6.1: 大規模データにおける Inv_score+LDA の実行時間と誤分類率

第7章　結論

7.1　まとめ

本研究では，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以下の４点について述べた．

クラスタリングを用いた高速化

クラスタリングを用いてテキスト群を振り分けることで，全ての組み合わせに対して対訳テキスト判定を行う必要が無くなり，また並列化も可能であることから，全対全比較では計算量が大きくなってしまう大規模なテキスト群に関しても高速に対訳テキスト抽出を施すことが可能となる．

実験結果として，サンプリングと k-medoid 法の比較を行ったところ k-medoid 法によるクラスタリングはテキスト集合の平均数は小さくなるが計算コストが大きくまたテキスト集合が小さくなることによる誤分類が発生してしまうため，サンプリングの方が計算コストの面からも精度の面からも良い手法であることが示された．

LDA による次元削減

LDA を用いた次元削減を行うことで，クラスタリングによる振り分けの精度に本質的に影響が無いような単語を削減した．

LDA で抽出した単語のみを使ってクラスタリングを行ったところ，振り分けの精度はほぼ変わらずに計算速度が 2 倍以上速くなっているのが示された．

転置インデクスによる高速化

転置インデクスを用いた対訳テキスト検索をクラスタリングに応用することで，特に振り分けにかかるコストを削減することが出来た．また LDA と組み合わせることで更に良い性能が示されることが確認された．

計算コストと精度の関係の測定を行い，誤分類率を 5 %程度に抑えた場合で，振り分けの速度が 6 倍程度，全体としての処理速度が 2 倍程度速くなっていることが示された．

大規模データへの適用

提案手法の，大規模データに対しての有効性を検証した．100,000 テキストに対して提案手法を適用したところ，全体としての処理速度が 2.8 倍程度速くなっていることが示された．これは大規模データの方がクラスタリングによるコスト削減の効果が大きいことを示していると思われる．

これらの手法を組み合わせることで，大量のテキストに対しての対訳テキスト抽出を高速化した．クラスタリングを行うことで単純な全対全比較では困難だった並列化が容易に可能であり，また LDA と転置インデクスを用いることでサンプリングにかかる時間を削減することが出来た．大規模なテキストコーパスからの対訳テキスト抽出の高速化を考えた場合に，テキストを小さい集合に振り分ける方向と，振り分け後に対訳テキストを抽出する方向，両方の観点から高速化を適用したことになる．我々の提案手法により，これまで適用が困難だった量のテキストに対しても対訳テキスト抽出が施すことが可能になると期待出来る．

7.2 今後の課題

今後の課題として，次のような点が挙げられる．

対訳判定精度の向上

本研究では，テキスト群に対して本来対訳であるテキストを同じクラスタになるように振り分ける手法を提案したが，振り分け後に対訳判定の精度に関しては trans_score の精度に依存している．この部分の精度を向上させることでより精密に多くの対訳テキストを抽出できる事が期待される．

振り分け精度の向上

本研究で提案した振り分け手法は，振り分け数や多重度を調整することで精度を維持しつつ計算速度の高速化を狙ったものである．k-medoid 法はあまり良い結果が出なかったが，より高速で精度の高いクラスタリング手法を適用することで，更に良い結果が得られることが期待される．

対訳例の増加

本研究では Fry のコーパスのニュース記事を対象として対訳テキスト抽出を行ったが，世の中には他にも物語や政府公式文書など様々な形式の対訳が存在しうる．これらの対訳テキストに対して本手法及び trans_score を適用することで，汎用性に関する検討が可能であると思われる．

テキスト数の増加

本研究では 100,000 程度のテキストを対象に評価実験を行った．しかし実際の Web 空間には数千万・数億レベルの非常に大量のテキストが存在するため更なる大規模なテキストに対して手法の評価を行う必要があると思われる．

Web テキストからの対訳テキスト抽出

本手法は Web テキストに対しての適用を考慮してはいるが，対訳テキストは本研究の範囲では Fry のコーパスのニュース記事を対訳判定の対象としており，実際に非常に大量の Web 文書からの未知の対訳テキストの抽出は行っていない．日夜増え続ける Web 上の対訳テキストが収集することが最終的な目標である．

Web クローラとの統合

Web ページからリンクを抽出してリンク先を収集する一連の流れを繰り返すクローラに対して対訳テキスト抽出手法を組み込むことで，クローラで集めるテキストから自動で対訳テキストを抽出することが可能となる．これにより，様々な目的で利用されるクローラに対して副次的に対訳テキストを抽出することが可能となり，結果として大量の対訳テキストが集まることが期待される．

参考文献

[1] M. Adriani. English-dutch CLIR using query translation techniques. In *CLEF '01: Revised Papers from the Second Workshop of the Cross-Language Evaluation Forum on Evaluation of Cross-Language Information Retrieval Systems*, pp. 219–225, London, UK, 2002. Springer-Verlag.

[2] E. Aramaki, S. Kurohashi, H. Kashioka, and H. Tanaka. Probabilistic model for example-based machine translation. In *MT Summit X*, pp. 219–226, 2005.

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

[4] P.F. Brown, S.D.Pietra, V.J.D. Pietra, and R.L. Mercer. The mathematic of statistical machine translation: Parameter estimation. *Computational Linguistics*, Vol. 19, No. 2, pp. 263–311, 1994.

[5] J. Chen, R. Chau, and C.H. Yeh. Discovering parallel text from the World Wide Web. In *ACSW Frontiers '04: Proceedings of the second workshop on Australasian information security, Data Mining and Web Intelligence, and Software Internationalisation*, pp. 157–161, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc.

[6] 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.

[7] EDR 電子化辞書. `http://www2.nict.go.jp/kk/e416/EDR/J_index.html`.

[8] M. Franz, J.S. McCarley, and S. Roukos. Ad hoc and multilingual information retrieval at IBM. In *Text REtrieval Conference*, pp. 104–115, 1998.

[9] 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.

[10] 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.

[11] T. Hofmann. Probabilistic Latent Semantic Indexing. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 50–57, August 1999.

[12] L. Kaufman and P.J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons inc, 1990.

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

[14] J.B. Macqueen. Some methods of classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathemtical Statistics and Probability*, pp. 281–297, 1967.

[15] R.T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In Jorgeesh Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *20th International Conference on Very Large Data Bases, September 12–15, 1994, Santiago, Chile proceedings*, pp. 144–155, Los Altos, CA 94022, USA, 1994. Morgan Kaufmann Publishers.

[16] R.T. Ng and J. Han. CLARANS: A method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 14, No. 5, pp. 1003–1016, 2002.

[17] K. Paik, S. Satoshi, and H. Nakaiwa. Automatic construction of a transfer dictionary considering directionality. In *COLING 2004 (The 20th International Conference on Computational Linguistics)*, pp. 31–38, 8 2004.

[18] P. Resnik. Mining the web for bilingual text. In *Proceedings of the International Conference of the Association of Computational Linguistics*, Maryland, 1999.

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

[20] 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.

[21] K. Tsuji and K. Kageura. Automatic generation of japanese english bilingual thesauri based on bilingual corpora. *J. Am. Soc. Inf. Sci. Technol.*, Vol. 57, No. 7, pp. 891–906, 2006.

[22] Webstemmer. `http://www.unixuser.org/~euske/python/webstemmer/index.html`.

[23] Justin Zobel, Alistair Moffat, and Kotagiri Ramamohanarao. Inverted files versus signature files for text indexing. *ACM Transactions on Database Systems*, Vol. 23, No. 4, pp. 453–490, 1998.

[24] 宇津呂武仁, 日野浩平, 堀内貴司, 中川聖一. 日英関連報道記事を用いた訳語対応推定. 自然言語処理, Vol. 12, No. 5, pp. 43–69, 2005.

[25] 福島健一. Web コーパスからの日英対訳テキストの抽出. 東京大学, 学士論文, 2006.

発表文献

[1] 斎藤大, 吉田慎一郎, 田浦健次朗, 近山隆. Web 上の対訳テキストの大規模高速抽出手法. 第 13 回言語処理学会年次大会併設ワークショップ, 3 2007.

[2] 斎藤大, 田浦健次朗, 近山隆. 大量の Web 文書からの対訳テキスト抽出手法. 日本ソフトウェア科学会 第 24 回大会, 9 2007.

[3] 斎藤大, 田浦健次朗, 近山隆. 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. 第 14 回言語処理学会年次大会, 3 2008 (発表予定).

謝辞

本研究を進めるにあたり，大変多くの方にお世話になりました．

近山隆教授には，論文作成やプレゼンテーションなど要所で的確なアドバイスを頂きました．個々の論文に関することから研究者としての考え方まで様々な面で多くの知識を伝授して頂きました．田浦健次朗准教授には，研究の方針から論文の書き方まで熱心にご指導して頂きました．また研究に行き詰った時は相談に乗って頂き先行く道を開いてくださるなど，学部時代からの３年間大変お世話になりました．

博士課程の三輪誠さんには，研究のことや論文の書き方・プライベートのことまで日頃から多くの助言を頂きました．またミーティングではいつも鋭い指摘をして頂きました．近山・田浦研究室の卒論生で同期だった福島健一君には，普段から勉強会や学会で様々な議論をさせて頂き，多くの刺激を受けました．福島君の研究が無ければ本研究を進めることも出来ませんでした．

新領域の同期の皆様，特に渡部克也君，下村直也君，杉谷心君，瀧口智香湖さんには専門が違うからこその鋭い指摘を頂き，研究を進める上で参考になる意見を頂きました．近山・田浦研究室の皆様，特に関谷岳史君，坂本祥吾君，吉田慎一郎君とは卒論生の時から一緒に議論や研究をしやすい環境を作って頂きました．他の近山・田浦研究室の皆様にも公私にわたり多くの助言を頂きました．

ここに，心より感謝の意を表します．

平成 20 年 1 月 29 日