MapReduce を用いた分布類似度計算による関連語判定

田中 慎平† 岡部 正幸†† 梅村 恭司‡

†‡豊橋技術科学大学 情報・知能工学系 〒441-8580 愛知県豊橋市天伯町雲雀ヶ丘 1-1

††豊橋技術科学大学 情報メディア基盤センター 〒441-8580 愛知県豊橋市天伯町雲雀ヶ丘 1-1

E-mail: †tanaka@ss.cs.tut.ac.jp, ††okabe@imc.tut.ac.jp, ‡umemura@tut.jp

あらまし シソーラスを文書集合から構築するシステムの一つに山本らのシソーラス構築システムがある．しかし，このシステムには処理時間が長いという問題がある．本論文ではこの問題を解決するために MapReduce の導入を考え，高速化に成功したことを報告する．

キーワード シソーラス，MapReduce，分布類似度

Related Terms Judgment by Distributional Similarity that uses MapReduce

Shinpei Tanaka† Masayuki Okabe†† Kyoji Umemura‡

†‡Department of Computer Science and Engineering, Toyohashi University of Technology

††Information and Media Center, Toyohashi University of Technology

1-1 Hibarigaoka, Tenpaku-chou, Toyohashi-shi, Aichi, 441-8580 Japan

E-mail: †tanaka@ss.cs.tut.ac.jp, ††okabe@imc.tut.ac.jp, ‡umemura@tut.jp

Abstract Thesaurus building system by Yamamoto and et al. builds the thesaurus from document set. This system, however, has problem that its processing time is long. In this paper, we report that MapReduce makes the processing time short.

Keyword Thesaurus, MapReduce, Distributional Similarity

1 はじめに

自然言語処理において，語彙に関する知識は重要な要素である．中でも，類義語に関する知識は検索や翻訳等の処理において重要な役割を果たすため，類義語の自動獲得や獲得した情報の活用方法,シソーラス(類義語辞書）の整備といった関連研究は盛んに行われてきた．

そういった関連研究の一つに，山本らによって提案されたシソーラス構築システム[9]がある．このシステムは自然文テキスト集合から関連語対のリスト（シソーラス）を自動構築するというシステムであるが，シソーラス構築の全処理工程において辞書を一切使用しない，故に辞書に登録されていない未知語にも対応できるという特徴的な長所を持つ反面，精度が低い，得られるシソーラスの量が少ない，処理時間が長いという短所が存在した．現在では，この内精度が低い，得られるシソーラスの量が少ないという短所に対しては，當間らによる研究[8]を用いた変更を経て改善されているが，処理時間が長いという短所は依然残っている状態である．処理時間の長さは，研究効率の低下やシステム大規模化の阻害といった問題にも繋がってしまうため，改善の必要性は高いと考えられる．

そこで本研究では，この処理時間が長いという短所を改善する方法として分散処理の導入を考えた．これは Pantel らの発表した論文[6]に記載されている事実から，シソーラス構築システムにおいて最も処理時間の長い処理である分布類似度の計算を高速化できるのではないかと考えたからである．Pantel らの論文によれば，分布類似度の計算は分散処理用のプログラミングモデル MapReduce でも実装することが可能であり，高速に動作するとある．Pantel らの論文とシソーラス構築システムとでは，分布類似度の定義式こそ異なるものの計算に利用するパラメータには共通する部分があるため，シソーラス構築システムでも同様に高速化が可能だと判断し，MapReduce の導入を実行した．

本論文では，シソーラス構築システムにおける分布類似度の計算を分散処理用のプログラミングモデル MapReduce で実装した結果，処理時間の大幅な軽減が見られたこと，それから副次的な効果として水平スケーラビリティ性が得られたことを報告する．

2　解説

2.1　分布類似度

「意味的に似ている語句は，その出現文脈の分布も似ている傾向がある」という考え（分布仮説[3]）に基づいて提案された，語と語の類似性を測る概念を分布類似度という．分布類似度の考えを利用することで，出現文脈の分布の類似度から逆に語句の間の類似度を機械的に推定することができる．

分布類似度を用いた手法による知識獲得には，大量に入手可能な生コーパスを知識源として利用できるという利点がある反面，類義語と同義語を区別する方法が今の所報告されていないこと，計算量が大きくなりがちであることなどの問題点も存在する．

分布類似度に関しては文献[5]がより詳しい．

2.2　MapReduce

MapReduce とは分散処理のためのプログラミングモデルである．MapReduce はサーバやインターネットにある膨大な情報を分散並列処理によって効率良く処理させるために，Google によって考案された[2]．しかし Google からは MapReduce の実装は公開されていないため，実際に MapReduce を利用する場合は[2]を元に作成された分散処理プラットフォーム Hadoop[4]を利用する必要がある．Hadoop が提供する MapReduce の実装である，Hadoop/MapReduce の仕様に従ってプログラミングすることで，分散並列処理を行うアプリケーションを作成することができる．

MapReduce の処理は，Map フェーズと Reduce フェーズの 2 段階に分けて実行される仕組みになっている．Map フェーズは，処理対象の入力データを細かいブロックに分割し，多数のサーバに分散して処理を行うフェーズである．このフェーズでは Map タスクと呼ばれる，データを分解し，必要な情報を抽出し，有用な形へと変換し出力する，いわゆるフィルターの役割を果たす処理が行われる．Reduce フェーズは，Map フェーズでの処理結果を集計するフェーズである．このフェーズでは Reduce タスクと呼ばれる，抽出された情報を集約する処理が行われる．

2.3　シソーラス構築システム

本研究で扱うシソーラス構築システムは，次の 3 工程からシソーラスを構築するシステムである．

①　単語の切り出し
②　候補対の選出
③　関連語の判定

以下にそれぞれの工程について順に説明する．

①　単語の切り出し

第 1 工程ではコーパスから単語の切り出しを行う．ここでは武田のキーワード抽出アルゴリズム[7]を使用して単語の切り出しを行うが，このシステムにおいて単語として扱うものは，「キーワード」と「文脈単語」の 2 つである．

「キーワード」とは武田のアルゴリズムによって抽出された，文脈を推定するのに有用だと考えられる語のことである．また，「文脈単語」とは武田のアルゴリズムによって抽出された，キーワードほど強い意味は無いが文脈上の意味を把握する上で有用な情報となる語のことである．

②　候補対の選出

第 2 工程では関連語の候補となる単語対を選出する．これは，第 1 工程において切り出された単語集合から考えうる単語対全てに対して関連関係にあるか調査する場合，計算量が問題となるため，関連語となる見込みの無い単語対をふるい落とす処理である．

③　関連語の判定

第 3 工程では新たに周囲単語対という概念を定義し，周囲単語対を文脈として捉えた場合の分布類似度から関連語の判定を行う．

周囲単語対の定義を，「単語 X に対して直前に現れた単語を α，直後に現れた単語を β とする時，ペア（α，β）を単語 X の周囲単語対とする」と定めた時，単語 X，単語 Y，周囲単語対の関係性は表 2-1 で表すことができる．この関係性は，単語 X と単語 Y を第 2 工程で得られた候補対に置き換えることで，候補対とその周囲単語対の関係に置き換えることができるため，周囲単語対を文脈として捉えた場合には，候補対とその出現文脈の関係ともみなすことができる．

よって表 2-1 に分布類似度として赤池情報量基準（AIC）[1]を用いた独立性検定を適用し，そのスコアから関連性の判定を行うのが第 3 工程での処理である．

独立性検定ではモデル M1「候補対の周囲単語対は独立である」と，モデル M2「候補対の周囲単語対は独立でない」を考え，表 2-1 から以下の式を用いて検定を行う．AIC_{M1} と AIC_{M2} の差が 1 より大きければ有意な差であるとみなすことができる．

$$AIC_{M1} = (-2)\left\{ a\log\frac{N_{ab}N_{ac}}{N} + d\log\frac{N_{cd}N_{bd}}{N} \right\} + 2\times 2$$

$$AIC_{M2} = (-2)\left\{ a\log\frac{a}{N} + d\log\frac{d}{N} \right\} + 2\times 3$$

したがって次のような関連語対集合 Relevants'が得られる．（ＸＹ：単語Ｘと単語Ｙのペア）

$$\text{Relevants'} = \left\{ XY \mid AIC_{M1} - AIC_{M2} > 1 \right\}$$

表 2-1 単語Ｘ，単語Ｙ，周囲単語対の関係性
（シソーラス構築システム）

	Y	$\overline{Y}$	計
X	a	b	N_{ab}
$\overline{X}$	c	d	N_{cd}
計	N_{ac}	N_{bd}	N

― 事象の定義 ―

X （$\overline{X}$）：単語 X の周囲単語対である（ない）

Y （$\overline{Y}$）：単語 Y の周囲単語対である（ない）

― パラメータの定義 ―

それぞれ定義に該当する周囲単語対の数を表す

$$a = \mid X \cap Y \mid, \quad b = \mid X \cap \overline{Y} \mid$$

$$c = \mid \overline{X} \cap Y \mid, \quad d = \mid \overline{X} \cap \overline{Y} \mid$$

$$N_{ab} = a + b, \quad N_{ac} = a + c$$

$$N_{bd} = b + d, \quad N_{cd} = c + d$$

$$N = a + b + c + d$$

3 MapReduce の導入

Pantel らの論文においては，単語 X が出現する文脈の種類数，単語 Y が出現する文脈の種類数，単語 X と Y に共通する文脈の種類数をパラメータとして用いて分布類似度の計算を行っている．対してシソーラス構築システムでは前述のように，表 2-1 に示す 9 つの値をパラメータとして利用する．表 2-1 における N_{ab} は単語 X が出現する文脈の種類数，N_{ac} は単語 Y が出現する文脈の種類数，a は単語 X と Y に共通する文脈の種類数にそれぞれ対応しているため，その３つのパラメータに関しては MapReduce による計算が可能であることは間違いないが，シソーラス構築システムの分布類似度の計算に MapReduce を導入するには，他のパラメータも求める必要がある．

この点について検討した結果，MapReduce でもパラメータ b,c,d が近似的に計算できることが分かり，シソーラス構築システムの分布類似度の計算を MapReduce を用いて実現することができた．

4 実験

MapReduce を導入した結果，システムの性能がどのように変化したかを確認するために処理時間の比較実験を行った．また，MapReduce 導入システムでは処理リソースの変化によって性能が変化するかを確認するための実験も行った．実験の詳細と結果を以下に示す．

4.1 実験環境

実験に使用したマシンのスペックを以下に示す．

OS：Ubuntu Desktop 10.4
CPU：AMD Athlon™ Ⅱ X4 635 Processor 2.9GHz 4 コア
メモリ：2GB
ディスク：1TB×1
LAN：Gigabit-Ethernet 接続

4.2 実験方法

○実験 1

処理対象のデータとして NTCIR1 の論文コーパス 1 万行～5 万行を用意．これに対してベースシステムと MapReduce を導入したシステムを用いてシソーラスを構築し，それぞれ分布類似度の計算にかかった時間を測定する．この実験では，ベースシステムは 1 台での処理時間を測定．MapReduce 導入システムは 6 台を使ったクラスタ上での処理時間を測定．ただし 1 台はクラスタの管理専用のため，実質処理リソースは 5 台分となる．

○実験 2

処理対象のデータとして NTCIR1 の論文コーパス 16 万行と 32 万行を用意．これに対して，MapReduce 導入システムが動いているクラスタを構成するノード（マシン）の台数を変化させた場合の処理時間を測定する．

4.3 実験結果

○実験 1

処理時間の測定結果を表 4-1 に示す．処理時間が順に $\frac{1}{5}$，$\frac{1}{35}$，$\frac{1}{124}$ に減少し，大幅な高速化に成功していることが分かる．

表 4-1　処理時間測定結果

	入力テキスト（行）		
	10000	30000	50000
ベースシステム	446	3177	11049
MapReduce 導入システム	91	91	89

※単位は秒

○実験 2

処理時間の測定結果を図 4-1 に示す．処理台数の増加に伴って処理時間が減少している様子が見て取れる．

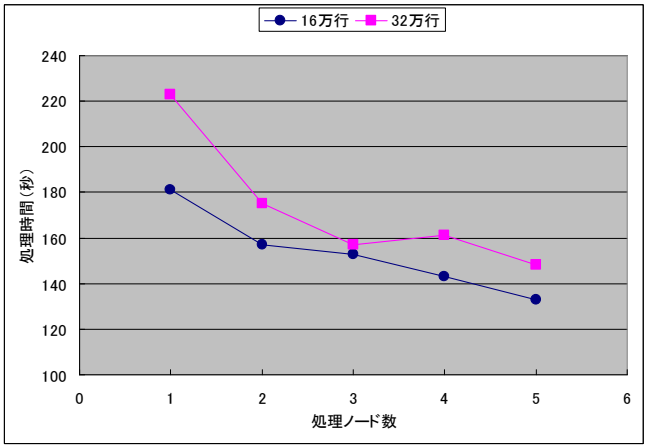

図 4-1　処理ノード数を変化させた場合の処理時間

5　考察

まず表 4-1 の結果から，当初の目的であった処理時間の長さという問題は解決できたといえる．一部処理時間が処理リソースの増加以上に減少しているが，これは近似処理による計算量の減少によるものだと考えている．これが実際にどの程度処理時間の短縮に寄与しているかは今後の研究によって明らかにする予定．それから，分布類似度に使用したパラメータが近似的なものだったことが理由だと考えられるが，システムの出力として得られるシソーラスに差異が見られた．分布類似度のスコアが大きい単語対上位 100 件を比較した場合に，入力データが 5 万行で 78%の一致，8 万行で 93%，16 万行で 99%と，入力データが大きくなるにつれて影響は小さくなっていくようだが，この点については今後改善する必要があると考えている．

次に図 4-1 の結果から，分布類似度の計算に限っての話ではあるが，スケーラビリティ性があることが分かる．今回の実験で使用した Hadoop の特性から，MapReduce での実行が正常に行われるならばスケーラビリティが確保されることは予測されていたので，実際にデータとして取れたことで MapReduce での実行が正常に動作していることが分かる．

6　まとめ

本論文では，シソーラス構築システムにおける分布類似度の計算を分散処理用のプログラミングモデル MapReduce で実装した結果，処理時間の大幅な軽減が見られたこと，それから副次的な効果として水平スケーラビリティ性が得られたことを報告した．

また，得られた結果についても，先行研究との比較を行った．

7　参考文献

[1] 赤池弘次，甘利俊一,北川源四郎,樺島祥介,下平英寿．赤池情報量基準 AIC．共立出版, 2007.

[2] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. OSDI'04 : Sixth Symposium on Operating System Design and implementation, 2004.

[3] Zellig S. Harris. Distributional structure. Word, Vol. 10, pp. 146-162, 1954.

[4] Hadoop homepage. http://hadoop.apache.org/ (2011/01/06 確認).

[5] Dekang Lin. Automatic retrieval and clustering of similar words, Proc. Of the COLING/ACL 1998, pp. 786-774, 1998.

[6] Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu and Vishnu Vyas. 2009. Web-Scale Distributional Similarity and Entity Set Expansion. In Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-09). pp. 938-947. Singapore, Singapore.

[7] 武田 善行，梅村 恭司．キーワード抽出を実現する文書頻度分析．計量国語学，Vol.23, No.2, pp.65-90, September 2001.

[8] 當間 雅，梅村 恭司．語の出現類似性のための統計的モデルとシソーラス構築への適用．言語処理学会第 13 年次大会.

[9] Eiko Yamamoto and Kyoji Umemura. Related Word-pairs Extraction without Dictionaries, LREC-2004 pp.1309-1312, 2004.