

Hadoop による分布類似度計算の分散処理

田中 慎平[†] 梅村 恭司[†] 岡部 正幸^{††}

[†]豊橋技術科学大学 情報・知能工学系

^{††}豊橋技術科学大学 情報メディア基盤センター

1. はじめに

「言語は曖昧性であふれている」(参考文献[1]より引用) 上述の論文にあるように、言語には同じ情報に対して多くの表現が存在し、また同じ表現に対して多くの解釈が存在する。この言語の曖昧性に関する問題は、多くの場合、人間にとっては易しいが計算機にとっては難しい。この問題は自然言語処理における大きな課題となっており、多くの研究者がこの課題に取り組んできた。

この課題に対して、「同義・含意関係知識の獲得」というアプローチがある。これは語の同義関係(同じ意味を持つ)や含意関係(上位・下位関係にある)に関する知識を収集し、計算機に与えることで、言語の曖昧性に対処するというものである。このアプローチには幾つかの手法が存在するが、その内の一つに近年盛んに研究されている「分布類似度を用いた手法」がある。この手法は分布類似度を用いてテキストコーパスから同義表現を収集するというものであるが、大量に入手できる生コーパスをそのまま利用できるという利点がある反面、計算量が大きくなりがちであり、処理に時間がかかるという問題がある。

本稿ではこの問題を解決する方法として、分散並列処理用のプログラムモデル MapReduce による分布類似度計算を提案する。MapReduce でどのように実装するのか、結果として処理時間はどうなるのか、以下に報告する。

2. 解説

本章では本研究で扱う技術について簡単に解説する。

2.1 分布類似度

「意味の似ている単語は同じような使われ方をする」という傾向は経験則として知られているが、この傾向は Harris によって分布仮説[4]としてまとめられている。そしてこの分布仮説によれば「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」となっている。

分布類似度[5]とは、この分布仮説に基づいて提案された「語と語の類似性を測る概念」である。分布仮説の考えを逆に捉えることで、出現文脈の分布の類似性から語句の間の類似性を推定する。分布を比較し、定量的に評価することで得られる値を類似性の度合いとするため、分布類似度と呼ばれる。

分布類似度の算出には、分布から得られる統計的なパラメータと、そのパラメータを変換する式が必要となる。パラメータと変換式について、以下に例を示し解説する。

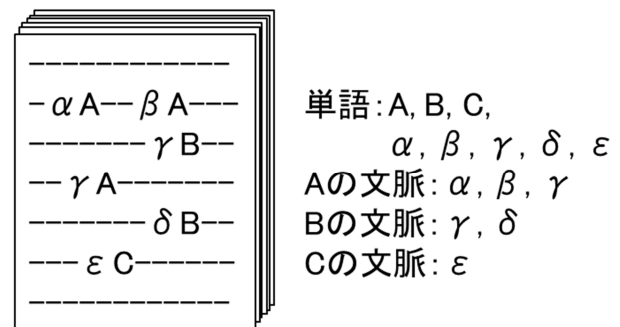


図1 テキスト例

図1左に示すテキストが与えられた時、「単語前方に出現する1単語」を文脈と定義すると、単語A, B, Cとその文脈について図1右に示す関係性が得られる。そしてこの関係性を、文脈の数に着目して単語AとBについてまとめると、表1のようになる。

表1 単語Aと単語Bの関係性

	○:Bの文脈	×:Bの文脈	合計
○:Aの文脈	1	2	3
×:Aの文脈	1	1	2
合計	2	3	5

この時、表1に示す9つの値が、単語AとBの出現文脈の分布から得られる値であり、分布類似度の計算に利用できるパラメータである。表1を一般化したものを表2に示す。

表2 分布類似度の計算に利用できるパラメータ

	Y	\bar{Y}	合計
X	a	b	N_{ab}
\bar{X}	c	d	N_{cd}
合計	N_{ac}	N_{bd}	N

— 事象の定義 —

単語 X (\bar{X}) : 単語 X の文脈である (ない)

単語 Y (\bar{Y}) : 単語 Y の文脈である (ない)

— パラメータの定義 —

$$a = |X \cap Y|, b = |X \cap \bar{Y}|, c = |\bar{X} \cap Y|, d = |\bar{X} \cap \bar{Y}|$$

$$N_{ab} = a + b, N_{ac} = a + c, N_{bd} = b + d, N_{cd} = c + d$$

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

分布類似度の計算に利用できるパラメータは表 2 以外にもあるが、よく使われるものがこの 9 つである。特にパラメータ a は共起頻度と呼ばれ、後述する変換式の多くに利用されている。

パラメータを変換する式については、明確な定義式は存在しない。分布から得られるパラメータを用いて一意な値を算出する式であれば、類似度としての精度はともかく、変換式として利用できる。関連研究においては以下に示すコサイン関数やダイス相関係数がよく使われている。

$$\text{コサイン関数} \quad f = \frac{a}{\sqrt{N_{ab} \times N_{ac}}}$$

$$\text{ダイス相関係数} \quad f = \frac{2a}{N_{ab} + N_{ac}}$$

2.2 MapReduce

MapReduce とは分散並列処理のためのプログラムモデルである。大規模なデータを効率よく処理するためのプログラムモデルとして、2004年に Google から発表された[2]。MapReduce はプログラムに制限を課し、プログラミングの自由度を下げることを代償にして、分散並列化に特化させている点に特徴がある。

MapReduce は Map・Shuffle・Reduce の 3 つの概念によって構成され、それぞれ以下のように定義されている。

- Map
入力データのレコード(行)毎に何らかの処理をする。処理はレコード毎に完結し、他レコードに依存しない。結果は「key」と「value」の組で表現する。
- Shuffle
Map の出力を「key」単位で集約する。
- Reduce
Shuffle で集約した「key」毎に何らかの処理をする。処理は「key」毎に完結し、他「key」に依存しない。結果は「key」と「value」の組で表現する。

Map での処理をレコード毎に完結する処理とすることで

他のレコードに依存しない状態を作り、最大でレコード数と同じ数の並列化を可能とする。また、出力を「key」と「value」という表現に限定することによって、Reduce においても最大で「key」の数と同じ数の並列化を可能とする。これらの制限によって、プログラミングの自由度は下がるが、分散並列化に特化させている。

MapReduce プログラミングにおいては、上記の定義に従いつつ Map・Shuffle・Reduce においてどんな処理をするかを定める。MapReduce でのワードカウントを例にして以下に解説する。

○ Map での処理内容

入力をスペースで区切り、key に単語、value に 1 を代入して出力する。

入力		Map	Map出力	
key	value		key	value
1:Hi	Ho		Hi	1
2:Hi			Ho	1
3:Ho			Hi	1
			Ho	1

○ Shuffle での処理内容

key が一致するレコードの value を連結する。

Map出力		Shuffle	Shuffle出力	
key	value		key	value
Hi	1		Hi	1,1
Ho	1		Ho	1,1
Hi	1			
Ho	1			

○ Reduce での処理内容

value を取り出して加算し、key はそのまま、value に加算された値を代入して出力する。

Shuffle出力		Reduce	Reduce出力	
key	value		key	value
Hi	1,1		Hi	2
Ho	1,1		Ho	2

MapReduce を使わずシンプルにワードカウントを行う場合は、配列を用意し、単語の格納と数値の加算を行えば良い。しかし、MapReduce の場合は他レコードに依存してはいけないために、複数レコードに跨る配列は使えず、上記のように処理を行う必要がある。

2.3 Hadoop

Hadoop[3]とはMapReduceを実装した分散並列処理用のソフトウェア・プラットフォームである。GoogleはMapReduceを発表したが、その実装は公開されていない。よってMapReduceを利用する場合はHadoopを利用する必要がある。

Hadoopにおいては、Hadoop MapReduceというフレームワークによってMapReduceの実装が提供されるため、プログラミングはMapとReduceの処理内容を記述するだけで良い。Shuffleは完全に実装されており、デフォルトでは前述のワードカウントの例と同様にvalueが連結されるようになっている。

3. 分布類似度の計算

本章では2.1節表2に示した9つのパラメータを用いた分布類似度計算のアルゴリズムについて解説する。ただし、分布類似度はパラメータさえ求まれば変換式に代入するだけなので、解説はパラメータを算出するまでとする。

3.1 シンプルな分布類似度計算

まずMapReduceを使わないシンプルな分布類似度計算のアルゴリズムについて解説する。

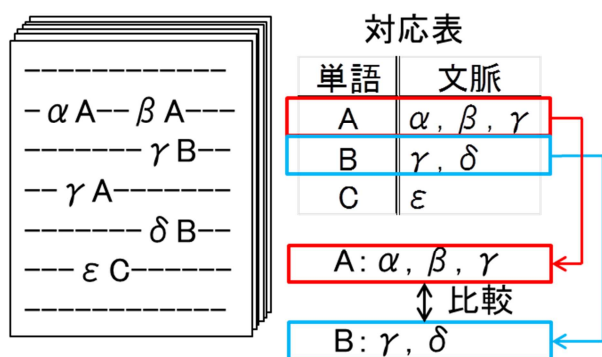


図2 シンプルなアルゴリズム

図2左に示すテキストが与えられた場合、まずテキストを全て読み込み、単語と文脈の対応表を作成する。そして対応表が完成したら2つの単語を取り出し、文脈を比較する。この時、一致する文脈の数がパラメータ a 、単語Aの文脈の数がパラメータ N_{ab} 、単語Bの文脈の数がパラメータ N_{ac} 、対応表の文脈の総数（重複は除く）がパラメータ N である。この場合は順に1, 3, 2, 5となっており、これは表1の値と一致する。そして表2のパラメータの関係性から4つの値が求まれば他の5つの値も求めることができるため、これで単語AとBの組み合わせに対する分布類似度計算に必要なパラメータが求まったことになる。

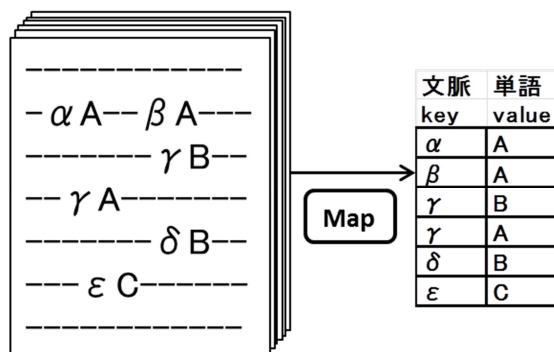
後は全ての単語の組み合わせについて同じ処理を繰り返すことで、テキストに出現する単語全ての組み合わせについて分布類似度が算出できる。

3.2 MapReduce 版分布類似度計算

次にMapReduce版分布類似度計算のアルゴリズムについて解説する。MapReduceでは他レコードに依存してはいけないうえに、シンプルなアルゴリズムのように対応表を使うことができない。よって別の方法で2単語の組み合わせに対する9つのパラメータを求める必要がある。本手法では3回のMapReduceによってパラメータを求める。（なお、以下の例ではShuffleは自動的に行われるとする）

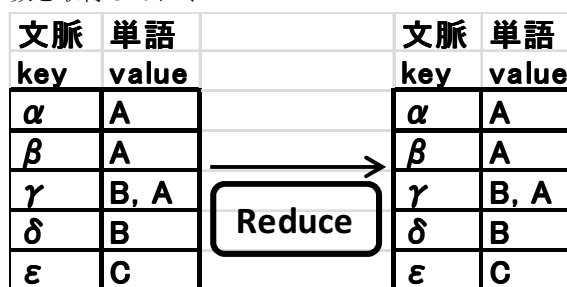
○ 1回目のMap

入力を分割し、keyに文脈、valueに単語を代入して出力する。



○ 1回目のReduce

valueの重複を省く。しかしこの例では重複は存在しないためそのまま出力される。また、Reduce出力のレコード数がパラメータ N と等価なため、レコード数を取得しておく。



○ 2回目のMap

keyとvalueを逆転させる。その際valueが2つ以上存在するならば、考え得る組み合わせを全て作って新しいkeyとする。（単語の組み合わせは=で表記）また、文脈の文字情報はもう必要がないため1に変換する。

文脈	単語		単語	
key	value		key	value
α	A		A	1
β	A		A	1
γ	B, A		B	1
δ	B		B=A	1
ϵ	C		A	1
			B	1
			C	1

Map

○ 2 回目の Reduce

value を加算する. key が単語の組み合わせの場合のみ, 先頭の単語と同じ単語の key から value 値をもらい, 自身の value に追加する. 本来は他レコードに干渉できないが, Shuffle の自然順序順にソートされるという仕様によって, Reduce で B, B=A の順に処理されることは決まっているため, 処理の順序を利用してグローバル変数を介して値のコピーを受け渡す.

単語			単語	パラメータ
key	value		key	value
A	1, 1, 1		A	3
B	1, 1		B	2
B=A	1		B=A	1/2
C	1		C	1

Reduce

○ 3 回目の Map

単語の組み合わせの順序を逆転させる.

単語	パラメータ		単語	パラメータ
key	value		key	value
A	3		A	3
B	2		B	2
B=A	1/2		A=B	1/2
C	1		C	1

Map

○ 3 回目の Reduce

2 回目の Reduce と同様にしてパラメータを追加する. 1 回目の Reduce から得たパラメータ N も追加する.

単語	パラ		単語	パラメータ
key	value		key	value
A	3		A	3
A=B	1/2		A=B	1/2/3/5
B	2		B	2
C	1		C	1

Reduce

このようにしてパラメータを求める. 単語の組み合わせ A=B の value は 1, 2, 3, 5 だが, これは順にパラメータ a, N_{ac} , N_{ab} , N であり, 表 1 と一致する.

4. 処理時間測定実験

Wikipedia から抽出した 1GB のテキストコーパスを用いて処理時間測定実験を行った. シンプルな分布類似度計算 (シングルプロセス) と MapReduce 版分布類似度計算 (16CPU での並列実行) の処理時間を測定した結果を表 3 に示す.

表 3 処理時間測定実験結果

	入力データ量		
	100MB	500MB	1000MB
シンプル[分]	54.5	546.3	1128.7
MapReduce 版[分]	1.8	2.3	3.3

MapReduce 版分布類似度計算の方が, 処理時間が大幅に短いことが分かる.

5. まとめ

本稿では分布類似度計算における処理時間の長さという問題に焦点を当て, 問題の解決方法として MapReduce というプログラムモデルの導入を提案した. 結果として処理時間の問題は改善したことを示し, MapReduce での実装方法を紹介した.

6. 参考文献

- [1] 乾健太郎. 自然言語処理と言い換え. 日本語学, Vol.26, No.11, pp.50-19, 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] Hadoop homepage. <http://hadoop.apache.org/> (2012/01/20 確認).
- [4] Zellig S. Harris. Distributional structure. Word, Vol. 10, pp. 146-162, 1954.
- [5] Dekang Lin. Automatic retrieval and clustering of similar words, Proc. Of the COLING/ACL 1998, pp. 786-774, 1998.