自然言語処理　144－18
(2001. 7. 17)

diff と言語処理

村田 真樹　　井佐原 均

独立行政法人 通信総合研究所
けいはんな情報通信融合研究センター
〒 619-0289 京都府相楽郡精華町光台 2-2-2
TEL:0774-95-2424　FAX:0774-95-2429　{murata,isahara}@crl.go.jp

あらまし

差分検出を行なう diff コマンドは言語処理の研究において役に立つ場面が数多く存在する．本稿では，diff を使った言語処理研究の具体的事例として，差分検出，データのマージ，書き換え規則の獲得，最適照合の例を示す．

キーワード　diff, mdiff, 差分検出，マージ，書き換え規則，最適照合

NLP using DIFF

Masaki Murata　　Hitoshi Isahara

Keihanna Human Info-communication Research Center,
Communications Research Laboratory
2-2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0289, Japan
TEL:+81-774-95-2424　FAX:+81-774-95-2429　{murata,isahara}@crl.go.jp

Abstract

Diff, a software program that detects differences between two sets of data, is often useful for natural language processing. This paper shows several examples of how *Diff* can be used in natural language processing. These examples include the detection of differences, the merging of two sets of different data, the extraction of rewriting rules, and the best matching of two different sets of data.

key words　Diff, Mdiff, Extraction of Differences, Merging, Rewriting Rules, Best Matching

1 はじめに

差分検出を行なう diff コマンドは言語処理の研究において役に立つ場面が数多く存在する．本稿では，まず簡単に diff の説明を行ない，その後，diff を使った言語処理研究の具体的事例として，差分検出，書き換え規則の獲得，データのマージ，最適照合の例を示す[1]．

2 diff と mdiff と mdiffc

本節では diff について説明する．本稿でいう diff とは UNIX のファイル比較ツール diff のことである．このコマンドは，与えられた二つのファイルの差分を順序情報を保持したまま行を単位として出力する[2]．例えば，

今日
学校へ
いく

ということが書いてあるファイルと

今日
大学へ
いく

ということが書いてあるファイルがあるとする．これらの diff をとると，差分の部分が

< 学校へ
> 大学へ

のような形で出力される．

ところで，diff コマンドには -D オプションという便利なオプションがある．これをつけて diff コマンドを使うと差分部分だけでなく共通部分も出力される．つまりファイルのマージが実現される．また，差分部分は C のプリプロセッサなどで使われる ifdef 文などで表現される．ここでは ifdef 文は見にくいので差分部分は以下のように表示することにする．

; ▼▼▼▼▼▼
(一つめのファイルにだけある部分)
; ●●●
(二つめのファイルにだけある部分)
; ▲▲▲▲▲▲

ここでは，"; ▼▼▼▼▼▼" は差分部分の始まりを，"; ▲▲▲▲▲▲" は差分部分の終りを意味し，"; ●●●" は差分を構成する二つのデータの境界を意味する．本稿では，-D オプションをつけてさらに ifdef の部分を上記のように表示して，ファイルのマージを行なう場合の diff を **mdiff**と呼ぶ (m は merge の m)．

実際に先ほどのデータに対して mdiff をかけてみると，以下のような結果になる．

今日
; ▼▼▼▼▼▼
学校へ
; ●●●
大学へ
; ▲▲▲▲▲▲
いく

「今日」が一致し，「学校へ」と「大学へ」が差分となり，「いく」がまた共通部分となっている．mdiff の出力は diff と異なり一致部分も出力されるためにわかりやすい．

また，mdiff の結果からは元の二つのファイルのデータを完全に復元することができる．共通部分と，差分部分の黒丸 (; ●●●) の上側だけを取り出すと，

今日
学校へ
いく

のように一つ目のファイルの情報が取り出される．また，共通部分と，差分部分の黒丸 (; ●●●) の下側だけを取り出すと，

今日
大学へ
いく

のように二つ目のファイルの情報が取り出される．このように元の情報を完全に復元できる．

また，mdiff では一致部分は片方のデータにあったものだけを表示し，不一致部分のみ両方のデータのものを表示するために，元の二つのデータよりもデータ量は削減できるが，上記のように元の情報を完全に復元できるために，復元できる状態でデータ量を削減するという意味で mdiff はデータ圧縮を実現しているものともいえる．

次に文字を単位とした mdiff を考える．言語処理の場合は文字単位で差分を取りたい場合が多い．そのようなときは一度ファイルの中身の情報を，一文字ずつ改行をして出力したファイルで mdiff をとればよい．例えば先のファイルの情報だと，

今

---

[1] 本稿の執筆は文献(1) で予告していた．

[2] diff コマンドの内部のアルゴリズムについては文献(2) の p.282 に説明がある．

日
学
校
へ
い
く

という形にしてから，mdiff をとればよい．本稿ではこの 1 文字単位で mdiff をかけることを mdiffc と呼ぶことにする (mdiffc の c は character の c)．

diff の表示は見にくく，mdiff は diff で表示される情報を完全に含むので以降の説明は midff を用いて行なう．以降の節では，実際にこの mdiff を使った言語処理の実例を見ていくこととする．

3 差分検出，および，書き換え規則の獲得

本節では mdiff を用いて差分検出したり，またその差分結果から書き換え規則を獲得する研究などを記述する．具体的には，以下のものを示す．

- 複数システムの出力の差分検出
- 差分の考察と書き換え規則の獲得

3.1 複数システムの出力の差分検出

筆者は以前，juman のシステムのバージョン[3,4]が複数乱立しているとき，この複数の juman の出力を mdiff によりマージして形態素結果の品質を向上させる[3]ようなことをしていた[4]．ここではこれを説明する．

「といったこと」を解析し，juman の A というバージョンの出力が

と　と　助詞
いった　言う　動詞
こと　こと　名詞

となっていて，B のバージョンの出力が

と　と　助詞
いった　行く　動詞
こと　こと　名詞

となっているとしよう．「いった」という語は「行く」と「言う」の曖昧性があり，B のバーションではこれを誤って「行く」の方の語であると出力していたとする．ここで midff をとると以下のような結果となる．

と　と　助詞
;▼▼▼▼▼▼
いった　言う　動詞
;●●●
いった　行く　動詞
;▲▲▲▲▲▲
こと　こと　名詞

mdiff をとることで複数のシステムの出力の差異を容易に検出することができる．この場合「いった」の部分が出力に差異があることがわかる．ここで，出力修正の作業者はこのような差分が検出された箇所においてどちらが正しいかを判断し，上が正しければなにもせず下が正しければ「;●●●」の先頭に “x” をつけるなどとすると決めておく．そのようにすると，“x” がなければ差分の下を，あれば差分の上の情報と区切り記号を消すことで，その作業結果のデータから自動的にそれぞれの差分からよい結果の方を選び，それぞれのバージョンのものよりも高い精度の結果を生成できる．また，差分の両方が誤っている場合がよくある．このときは「;●●●」の上の方のデータを実際に書き直すとよい．

この方法を用いると，修正できないものは両方のバージョンで同じように誤るものだけであり，多くの形態素誤りを修正できる．ここで注意すべきことは異なる性質のシステムを複数用意しないといけないということである．誤り方が同じシステムの場合だと多くの誤りを見逃すことになる．

また，システムが三つある場合は diff3 コマンドを使うとよい．diff3 は三つのファイルの差分を検出することができる．

上記では形態素解析を例にあげたが，他の解析でも解析結果を行単位にすることで mdiff で差分をとることができる．また，文字単位が必要ならば mdiffc を使えばよい．

ここでは，複数のシステムの出力の差分をとる話をしたが，一つをタグつきコーパスとし，それをなにかのシステムで解析した結果と比較することで，そのタグつきコーパスの誤りを検出し修正する[5]ということもできる．

3.2 差分の考察と書き換え規則の獲得

ここでは，文献(1) でも述べた話し言葉と書き言葉の diff の研究について記述する．この研究では，対応のと

[3] これは，文脈処理の研究(5) を行なう際に，文脈処理の前段階の形態素解析，構文解析の誤りを修正するために行なっていた．

[4] 同様の考え方は文献(6) にもある．また，この種の考え方がシステム融合 (複数のシステムを組み合わせることで個々のシステムの精度以上のものを得ることを目的としたもの) という形でよくしられている(7)．

[5] コーパス修正の先行研究には文献のもの(8, 9) がある．

表 1: 書き言葉データと話し言葉データの例

書き言葉データ	話し言葉データ
本	今日
論文	は
で	え
は	意味
意味	ソート
ソート	に
に	ついて
ついて	述べ
述べる	ます
。	一般に
一般に	ソート
ソート	って
は	いう
５０	の
音	は
順	だいたい

表 2: 書き言葉データと話し言葉データの diff の結果

;▼▼▼▼▼▼	ついて
本	;▼▼▼▼▼▼
論文	述べる
で	。
;●●●	;●●●
今日	述べ
;▲▲▲▲▲▲	ます
は	;▲▲▲▲▲▲
;▼▼▼▼▼▼	一般に
;●●●	ソート
え	;▼▼▼▼▼▼
;▲▲▲▲▲▲	;●●●
意味	って
ソート	いう
に	の
(右欄につづく)	;▲▲▲▲▲▲

表 3: 差分部分の抽出

書き言葉データ	話し言葉データ
本論文で	今日
	え
述べる。	述べます
	っていうの

れた話し言葉と書き言葉のデータを使い，それらの差分から話し言葉と書き言葉の違いを考察したり，話し言葉から書き言葉への言い換え規則，また，その逆のための規則を獲得した．データとしては，学会の口頭発表を話し言葉データとし，その口頭発表の内容が記されたその学会の予稿原稿を書き言葉として用いた．

例えば，話し言葉と書き言葉のデータが表1のような形で与えられたとする[6]．ここでは，差分がとりやすいように形態素解析システムなどで 1 行に 1 単語がはいるような形に変換してある．このような書き言葉と話し言葉のデータが与えられたとき，mdiff をとると，表2のような結果を得る．この結果から，差分部分だけを抽出すると表3のような結果が得られる．

この結果から，話し言葉には「え」などが挿入されること，また話し言葉では「っていうの」という表現をいれて発話をなめらかにすることなどがわかる．また，「述べる」が「述べます」と言い換えられることがわかる[7]．以上のように mdiff を使うことで話し言葉と書き言葉の差異を検出でき，またそれを考察することで，話し言葉と書き言葉の違いのようなものを調査できることがわかる．また，これらの差分は話し言葉と書き言葉の言い換え規則としてみることもできる．例えば，「え」の部分は，書き言葉になにもないところに話し言葉に変換する場合「え」をいれるという規則のように見ることができる．また，「述べる」と「述べます」の部分は，話し言葉に変換する場合は「述べる」を「述べます」に言い換える規則のように見ることができる．その意味で mdiff を用いることで言い換え規則，もしくは，変換規則のようなものを検出できることがわかる．

ここでは，話し言葉と書き言葉のデータを例にとった

[6] ここではわれわれの意味ソートの論文(10) のものを例にあげている．ところで，本稿では diff を扱ったが，その論文では UNIX の sort コマンドを用いて様々な情報を意味の情報でソートする，つまり，順序付けて並べるということを行ない，それらが種々の言語処理にどのように役に立つかを議論している．興味があれば，この文献も読むことをお奨めする．

[7] 実際には，ここであげた例ほどきれいに話し言葉と書き言葉は対応がとれず，「本論文は」と「今日」のような言い換え表現としてはよくない対応が多く，確率などを用いたソートを用いて確信度の高い良質な差分情報を集めるということを行なう (これについては文献(1, 11) を参照せよ)．

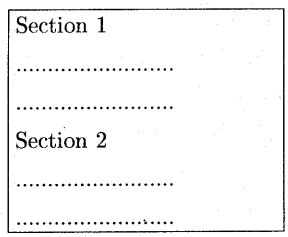

図 1: コーパスの構成

が，このようなことはさまざまなところで可能である．例えば，英文校閲前のテキストと英文校閲後のテキストで，mdiff をとると，どのような間違いをどのように直せばよいかがわかるし，また英文校閲用の規則のようなものが獲得できる．また，要約前のテキストと要約後のテキストで，mdiff をとると，どのように要約されているかを如実に見ることができるし，また要約用の規則のようなものが獲得できる[8]．その他にも対応のとれた性質の異なるデータに対して mdiff をとることで，さまざまな考察と，言い換え規則の獲得ができることだろう[9]．

4 データのマージ，および，最適照合

本節では mdiff のデータをマージする機能，および，そのマージの最適照合能力[10]を利用したものについて記述する．具体的には以下の三つについて記述する．

- 対訳コーパスの対応づけ
- 講演と予稿の対応づけ
- 最適照合能力を用いた質問応答システム

4.1 対訳コーパスの対応づけ

ここでは対訳コーパスの対応づけを考える[11]．ここで条件としてそれぞれのコーパスには対応する箇所に同じ記号が入っていることを前提とする．また，対応づけの単位はこの記号で区切られた部分であるとする．

例を図1にあげる．ここでは日本語のコーパスと英語の

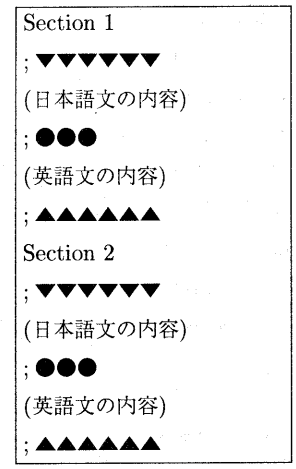

図 2: mdiff によって対応づけられた対訳コーパス

コーパスがばらばらに存在し，対応づけられていないとする．また，それぞれは図1のように両方とも Section 1 などの同じ形をしたセクション情報が与えられているとする．このとき，日本語と英語では同じセクションのものは同じ内容であるとする．この場合，これらのデータの mdiff をとることで，図2のような結果を得ることができる．この結果では，Section 1 などが共通部分となり，その他の部分が不一致部分となる．この不一致部分では日本語と英語が上下にわかれて格納されることになる．このようにすることで，mdiff を用いて対訳データが作成されることになる．

ここで示したものは，文ごとなどの細かい対応づけをするものでなく，セクションなどの大雑把なもので一見役に立たないように思えるかもしれないが，文の対応づけは難しい問題で，まずあらかじめ対応がとれていることがはっきりしている章，段落のレベルで対応づけをしてから細かい対応づけをするという考え方もあり，その意味ではこのような粗い対応づけも役に立つ．

また，ここで示したものは Section 1 などの情報を認識させて区分するだけなのでそのようなことをするプログラムを書くことでも同じように対訳データの対応づけを行なうことができる．しかし，mdiff を使うとそのようなプログラムも書くこともなく対応づけを容易に実現できるのである．

4.2 講演と予稿の対応づけ

本節では講演と予稿の対応づけ(16) を考える．この講演と予稿は，先の書き換え規則の獲得でも述べた書き言葉データと話し言葉データに対応する．講演は学会の口

[8] diff や mdiff は用いていないが，要約前のテキストと要約後のテキストで，DP マッチングを用い，どのように要約されているかを調べたり要約用の規則のようなものが獲得したりする研究として文献(12, 13) などがある．

[9] ここでの議論とは逆に，性質の同じ対応のとれたデータに対して mdiff をとることも考えられる．この場合，性質が同じデータのため，差分としては等価な表現対，つまり，同義表現のようなものが獲得されることになる．実際，われわれは異なる辞書の定義文を mdiff により照合することで，同義表現の獲得(11) を行なっている．

[10] diff の場合共通部分を最大にするような形で最適な照合を行なっている．

[11] 対訳コーパスは，機械翻訳(14, 15) の研究を行なう上で重要な研究資料となる．

<Chapter 1>
(1章の内容)
</Chapter 1>
<Chapter 2>
(2章の内容)
</Chapter 2>
<Chapter 3>
(3章の内容)
</Chapter 3>

図 3: 予稿データの構成

;▼▼▼▼▼▼
<Chapter 1>
(予稿のみの内容)
;●●●
(講演のみの内容)
;▲▲▲▲▲▲
(共通する内容)
;▼▼▼▼▼▼
(予稿のみの内容)
;●●●
(講演のみの内容)
;▲▲▲▲▲▲
(共通する内容)
;▼▼▼▼▼▼
</Chapter 1>
<Chapter 2>
(予稿のみの内容)
;●●●
(講演のみの内容)
;▲▲▲▲▲▲

図 4: 予稿と講演の mdiff の結果

頭発表で，予稿はその口頭発表に対応する論文のことである．このような講演と予稿が与えられたとき，講演の各部分と，予稿の各部分の対応がとれると，講演を聞いている時だと，それに対応する予稿の部分を参照できるし，予稿を読んでいるときだと，それに対応する講演の部分を参照できて便利である[16, 17]．本節ではこの講演と予稿の対応づけを mdiff で行なうことを考える．

ここでは特に予稿の各章が講演のどこの部分に対応するかを mdiff でもとめることにする．ここで予稿と講演とは話は同じ順序でなされると仮定する．また，予稿の

<Chapter 1>
(講演のみの内容)
(共通する内容)
(講演のみの内容)
</Chapter 1>
<Chapter 2>
(講演のみの内容)

図 5: 講演データへの章の情報の挿入結果

章が認識しやすいように予稿のデータには図3のように，“<Chapter 1>”のような記号を挿入しておく．この形にしておいて，予稿と講演のデータに対して，形態素解析をして各行に単語がくる状態で mdiff を使うことで，もしくは，mdiffc を使うことで，図4のような結果を得る．ここで，差分部分で予稿に対応する上半分の方を，“<Chapter 1>”のような記号を除いてすべて消し去ると図5のような結果を得る．図では元の講演のデータに対して“<Chapter 1>”のような記号だけが挿入された形になる．つまり，講演のどの部分が予稿のどの章にあたるかがわかることになる．

これは簡単にいうと，mdiff の照合能力を用いて予稿と講演を照合し，章の情報だけ残して予稿の情報を消し去ることにより，講演データに章の情報を挿入するということを行なっていることを意味する．このような予稿と講演の対応づけも mdiff を用いると簡単に行なえるのである[12]．

4.3 最適照合能力を用いた質問応答システム

本節では mdiff の最適照合能力を用いた質問応答システム[18, 19, 20, 21]について記述する．質問応答システムとは，例えば，「日本の首都はどこですか」と聞くと「東京」と答えそのものをずばり返すシステムである．知識が自然言語で書かれていると仮定すると，基本的には質問文と知識の文を照合し，その照合結果で疑問詞に対応するところを答えとして出力すればよい．例えば先の問題だと，「日本の首都は東京です」という文を探してきてこの文で疑問詞に対応する「東京」を解として出力するのである．ここではこれを mdiff で行なうことを

[12] この予稿と講演の対応づけを実際に文献[16]のデータで行なってみた．このときは，mdiff を用いまたデータは各行に単語がくるような状態で行なった．結果は文献[16]の精度と同程度か少しよい程度であった．mdiff を使うこのような簡単な処理でもこのような結果を得ることができるのである．

考える．

まず，質問文の疑問詞の部分を X に置き換え，また文末を平叙文に変換し，「日本の首都は X です」を得る．また，知識ベースから「日本の首都は東京です」を得る．ここでこの二つの mdiffc をとると以下のような結果を得る．

日
本
の
首
都
は
;▼▼▼▼▼▼
X
;●●●
東
京
;▲▲▲▲▲▲
で
す

ここで X と差分部分で組になっているものを解とすると，「東京」を正しく取り出せることになる．

ところで mdiffc を使う場合少々文に食い違いがあっても答えを正しく取り出すことができる．例えば，知識ベースの文が「日本国の首都は東京です」であったとする．この場合は mdiffc の結果は以下のようになる．

日
本
;▼▼▼▼▼▼
;●●●
国
;▲▲▲▲▲▲
の
首
都
は
;▼▼▼▼▼▼
X
;●●●
東
京
;▲▲▲▲▲▲
で
す

差分部分は少し増えるが X に対応する箇所は「東京」のままで，解を正しく抽出できる．

ところで，われわれが提案する質問応答システムでは類似度を尺度として用いた変形をくりかえし，質問文と知識データの文がより一致した状態で上記のような照合を行なう．このために類似度を定義する必要がある．mdiff を用いた場合は一致部分と不一致部分が認定できるので，類似度は (一致部分の文字数)/(全文字数) のような形で定義できる[13]．ここで，「日本国」と「日本」を言い換える規則があれば「日本の首都は X です」を「日本国の首都は X です」と言い換えて照合し，不一致部分を減らすことで，より確実に解を得ることができる[14]．

5 おわりに

本稿では diff を用いた言語処理の例をいくつか記述した．一つ，一つの話はそれほど珍しいものではないかもしれないが，diff に関係することでここまでいろいろな例をまとめたものはおそらくないだろう．他にも面白い利用方法があると思う．本稿であげた多数の例を参考にし，より面白い利用方法を考えて使うのもよいし，本稿であげた例と同じような使い方をしてもよい．diff を使って効率よく様々な研究がなされていくことを期待したい．

ところで，diff で行なうことは当然 DP マッチング (動的計画法による照合) などを丁寧にプログラミングすることでも実現できる．また，自分で細かくプログラミングした場合は，細かい調節もでき diff では行なえないことを行なえる場合もある．例えば，diff だと文字列的な比較しかできないが，自分でプログラミングした場合は品詞，意味情報なども含めて，差異の検出を行なったり，最適照合を行なうことができる．このため，精度を重視する場合，また，diff では扱えない細かい情報まで用いたい場合は地道に DP などをプログラミングした方がよい．

[13] ここでは mdiff により類似度を求めるようなことをしている．このように mdiff は文の類似性 / 類似度を求めることにも役に立つ．

[14] ここでは mdiff に基づく質問応答システムを述べたが，この mdiff に基づく質問応答システムは文献(21) でも述べているように文献(19) の研究の予備実験として構築したシステムである．精度の高いシステムを目指すならば，文献(19) にあるような構文情報なども用いたシステムを構築した方がよい．といっても mdiff を使うだけでも簡単な質問応答システムは容易に構築できることは簡便さの観点から価値がある．

しかし，システム開発ならばどうかはわからないが，研究の場合は簡便性が尊ばれる場合もある．また，簡便であるなら少し試してみるというのを気兼せずにできる．研究はいわば試行錯誤の賜物である．いろいろなことを試した結果，わかることもある．これが，簡便でなければ，あまり試さず新しい事実の発見が遅れることも十分考えられる．そういう意味では本稿のような diff を用いた簡便な言語処理も価値がある．

謝辞

本研究には 4.2節の話し言葉データと書き言葉データの対応づけの実験では独立行政法人通信総合研究所内元清貴研究員に実験データなどを提供してもらった．

参考文献

(1) 村田真樹, 井佐原均, 話し言葉と書き言葉の diff, ワークショップ「話し言葉の科学と工学」, (2001).

(2) 島内剛一, 有澤誠, 野下浩平, 浜田穂積, 伏見正則, アルゴリズム辞典, (共立出版株式会社, 1994).

(3) 松本裕治, 黒橋禎夫, 宇津呂武仁, 妙木裕, 長尾真, 日本語形態素解析システム JUMAN 使用説明書 version 1.0, (京都大学工学部 長尾研究室, 1992).

(4) 新谷研, 日本語形態素解析システムＪＵＭＡＮの精度向上, 京都大学工学部学士論文, (1995).

(5) 村田真樹, 日本語文章における名詞の指示対象の推定, 京都大学工学部修士論文, (1995).

(6) 石間衛, 藤井敦, 石川徹也, 日本語形態素・構文解析システム JEMONI の開発と評価について, 情報処理学会 自然言語処理研究会 98-NL-127, (1998).

(7) 村田真樹, 内山将夫, 内元清貴, 馬青, 井佐原均, 種々の機械学習手法を用いた多義解消実験, 電子情報通信学会言語理解とコミュニケーション研究会 NLC2001-2, (2001).

(8) 村田真樹, 内山将夫, 内元清貴, 馬青, 井佐原均, 決定リスト，用例ベース手法を用いたコーパス誤り検出・誤り訂正, 自然言語処理研究会 2000-NL-136, (2000), pp. 49–56.

(9) 村田真樹, 内山将夫, 内元清貴, 馬青, 井佐原均, 機械学習を用いた機械翻訳用モダリティコーパスの修正, 言語処理学会第 7 回年次大会, (2001).

(10) 村田真樹, 神崎享子, 内元清貴, 馬青, 井佐原均, 意味ソート msort — 意味的並べかえ手法による辞書の構築例とタグつきコーパスの作成例と情報提示システム例 —, 言語処理学会誌, Vol. 7, No. 1, (2000), pp. 51–66.

(11) 村田真樹, 井佐原均, 同義テキストの照合に基づくパラフレーズに関する知識の自動獲得, 自然言語処理研究会 2001-NL-142, (2001).

(12) 加藤直人, 浦谷則好, 局所的要約知識の自動獲得手法, 言語処理学会誌, Vol. 6, No. 7, (1999).

(13) 望主雅子, 荻野紫穂, 太田公子, 井佐原均, 重要文と要約の差異に基づく要約手法の調査, 情報処理学会自然言語処理研究会 2000-NL-135, (2000).

(14) 村田真樹, 馬青, 内元清貴, 井佐原均, サポートベクトルマシンを用いたテンス・アスペクト・モダリティの日英翻訳, 電子情報通信学会 言語理解とコミュニケーション研究会 NLC2000-78, (2001).

(15) 村田真樹, 馬青, 内元清貴, 井佐原均, 用例ベースによるテンス・アスペクト・モダリティの日英翻訳, 人工知能学会誌, Vol. 16, No. 1, (2001).

(16) 内元清貴, 野畑周, 太田公子, 村田真樹, 馬青, 井佐原均, 予稿とその講演書き起こしの対応付けおよび書き起こし のテキストのテキスト分割, 言語処理学会年次大会, (2001), pp. 317–321.

(17) 藤井敦, 伊藤克亘, 秋葉友良, 石川徹也, 音声言語データの構造化に基づく講演発表の自動要約話し言葉と書き言葉の diff, ワークショップ「話し言葉の科学と工学」, (2001).

(18) Masaki Murata, Masao Utiyama, and Hitoshi Isahara, Question answering system using syntactic information, (1999), http://xxx.lanl.gov/abs/cs.CL/9911006.

(19) 村田真樹, 内山将夫, 井佐原均, 類似度に基づく推論を用いた質問応答システム, 自然言語処理研究会 2000-NL-135, (2000), pp. 181–188.

(20) 村田真樹, 内山将夫, 井佐原均, 質問応答システムを用いた情報抽出, 言語処理学会第 6 回年次大会ワークショップ論文集, (2000), pp. 33–40.

(21) 村田真樹, 井佐原均, 言い換えの統一的モデル — 尺度に基づく変形の利用 —, 言語処理学会第 7 回年次大会ワークショップ論文集, (2001).