# 拡張ラグランジュ緩和を用いた同時自然言語解析法

鈴木 潤　　Kevin Duh　　永田 昌明
NTT コミュニケーション科学基礎研究所
{suzuki.jun, kevin.duh, nagata.masaaki}@lab.ntt.co.jp

## 1　はじめに

形態素解析，固有表現抽出，文節区切り，係り受け解析といった自然言語処理は，パイプライン方式で処理されることが現在一般的である．ここでいうパイプライン方式とは，形態素解析，係り受け解析といった各解析を独立かつ順番に処理する方式を指し，前の処理の結果が次の処理の入力となることを意味する．この処理方式がよく使われる背景として，従来，形態素解析，係り受け解析といった各解析処理は，それぞれ別の問題として扱って研究が進んできたことが大きな理由として考えられる．これは，研究の初期段階では，個々の問題自体がそれぞれ難しい問題であったため，個別に切り出して問題を定式化し，その問題に最も適した方法を用いて高い解析精度を得るという方法で研究を進めてきたことに起因すると思われる．しかし，ここ十数年の統計的手法の発達と適用により，多くの個別の自然言語解析問題は，高い水準で解析できる問題へと変化してきている．

また別の観点では，パイプライン処理の特性として挙げられる，先に処理された解析結果を利用して次の段階のモデル学習や解析処理を行う方法が現在多く利用されていることも，パイプライン処理が定着している理由の一つとして考えられる．例えば，係り受け解析といった個別の問題に着目した解析方法では，形態素解析した結果を受け取って解析処理が始まる．こういった場合には，素性などに前の処理結果を利用するため，前の処理が終わっていることが次の処理の前提となる．

このように，処理の利便性や素性設計の都合上，自然言語処理をパイプライン方式により処理することは，現段階で理にかなっていると言える．しかし，前の段階の処理がうまくいっていない時には，その解析誤りが次の処理へと蓄積されているという問題があることもよく知られている事実である．そこで，自然言語処理の次のステップとして，本稿では，従来パイプライン方式により処理されてきたいくつかの自然言語処理を，統合して同時に処理する方法に焦点をあてる．実際に近年では，同じモチベーションから多くの段階に分割された個々の自然言語解析問題を同時モデルで統合して解析する方法に関する研究が増えてきている．その中で，本稿では，文献 [1, 2] で紹介された DD-ADMM を用いて，様々な自然言語解析問題を統合して同時に解析するための定式化を提案する．本稿では，実際には，日本語の単語，文節，文分割および品詞付与の 4 つの自然言語処理問題を統合したモデルを紹介する．

## 2　自然言語処理の例

ここでは，$I$ 個の自然言語解析問題をパイプライン形式で逐次解析することを想定する．また，$\mathcal{X}_i$ を $i$ 番目の問題の入力空間，$\mathcal{Y}_i$ を $i$ 番目の問題の出力空間とする．このとき，従来のパイプライン方式では，以下の最適化問題を $I$ 回くり返し解くような形で処理が進められる．

$$\begin{aligned} \mathbf{y}_i^* &= \arg\max_{\mathbf{z}_i} f_i(\mathbf{x}_i, \mathbf{y}_i) \\ \text{w.r.t.} \quad & \mathbf{x}_i \in \mathcal{X}_i, \mathbf{y}_i \in \mathcal{Y}_i \end{aligned} \tag{1}$$

上式は，例えば $I$ 個の段階があるとした時の $i \in \{1, \ldots, I\}$ 番目の最適化問題を表している．パイプライン方式の場合は，$i-1$ 番目までに得られた $\mathbf{y}_j^*$ を，ただし $j \in \{1, \ldots, i-1\}$，$i$ 番目の入力 $\mathbf{x}_i$ の一部として利用することが多い．

次に，本稿で取り扱う，単語，文節，文分割と，品詞付与といった個別の問題を簡単に定義する．

### 2.1　単語分割

一般に「分かち書き」と呼ばれる処理である．本稿では，文節や文の場合と用語を統一するため，「単語分割」と呼ぶ．最も単純に問題の定式化を考えると，入力は文字列，出力は文字と文字の間が単語境界か，境界でないかを示すフラグで記述できる．文字間を分割する時を 1，しない時を 0 と表すとき，一つの入力文字間に対して一つの出力要素 $y_i = \{0, 1\}$ が対応する．また，入力文字列全体に対しては，入力文字数が $N$ のとき，出力空間 $\mathcal{Y}$ は，$\mathbf{y} \in \{[0, 1]\}^N$ の張る離散空間と定義できる．最適化自体は，式 (1) を用いる．

文献 [3] などに示されているように，崩れた文章や分野適用を考えた際には，このような簡単な方法で単語境界を決定するモデルが比較的効果的であることが示されている．

### 2.2　文節分割，文分割

文節や文分割も，本質的には，単語分割と同じ形式の問題である．よって単語分割と同様に出力空間間 $\mathcal{Y}$ は $\mathbf{y} \in \{[0, 1]\}^N$ の張る離散空間と定義できる．

ただし，例えば文節分割の場合，入力を文字列と仮定するか，単語分割の出力を入力と仮定する場合で入力は異なる．同様に，文分割の場合は，文字列，単語分割後の出力，文節分割後の出力といった，いくつかの可能性が考えられる．このとき，入力の違いによって変わるのは，使われる素性と，分割点の候補数の違いなどになる．よって，最終的には，最適化式は式 (1) となる．

### 2.3　品詞付与

単語の品詞を推定する問題は，日本語の場合単語境界も同時に推定しなくてはいけないことから，形態素解析として同時に処理されることが多い．本稿でも，基本的にこの形態素解析の処理方法を活用する．従来の形態素解析は，chasen や mecab[4] といった有名なツールでは，辞書を利用する．辞書エントリには，予め辞書エントリとエントリの連結成分にそれぞれコストが与えられており，コスト最小法に基づいて，最小コストのパスを求めることで，与えられた入力に対する出力を得る．つまり，ここでの品詞付与問題は，辞書エントリと連結成分により構成されたグラフに対し，始点から終点への最小コスト経路を探索する問題と等価であり，基本的には式 (1) で記述できる．

### 2.4　各問題の出力の構成要素

前述の 4 つの問題をまとめると，出力の構成要素は，品詞付与に出てくる辞書エントリとその連結成分，単語，文節，文分割の点推定モデルの大きく分けて合計 5 種類とみなすことができる．図 1 に，上記の構成要素の模式図を示す．本稿では，提案法の説明に，本図を用いて説明を行う．

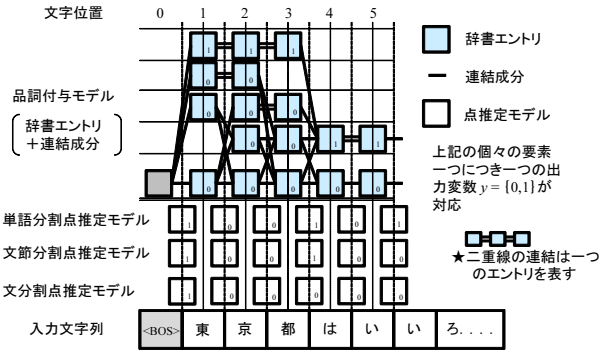


図 1: 各問題の出力要素の模式図

## 3 拡張ラグランジュ緩和に基づく双対分解 (DD-ADMM)

本稿では，文献 [1, 2] で提案された拡張ラグランジュ緩和による双対分解の枠組みに則って，複数の自然言語解析問題の同時モデルを定義する．提案法を説明するために，拡張ラグランジュ緩和による双対分解 (DD-ADMM) を簡単に説明する．

まず重要な点として，DD-ADMM では，出力は 0 または 1 の集合で表現されると仮定する．集合の要素が $N$ 個の時に，この集合を，全要素が $[0,1]$ の $N$ 次元のベクトルで表現する．このベクトルを，入力に対する出力ベクトルとする．

例えば，単語分割であれば，入力文字列の各文字間が単語分割になるかならないかを $[0,1]$ で表現できる．よって，文字列長個の $[0,1]$ 要素からなるベクトルが，一つの入力文字列に対する出力ベクトルと定義できる．また，品詞付与の問題であれば，各単語に対して，事前に設定した品詞候補から，出力として選択した一つを 1，それ以外を 0 とした $[0,1]$ ベクトル返す問題と考えることができる．よって，ラベル数 $L$ 個，単語数 $N$ 個としたとき，全体で $L \times N$ 個の $[0,1]$ 要素からなるベクトルが一つの入力に対する出力ベクトルと定義できる．ただし，この場合は，1 をとる要素数は出力ベクトル中，必ず文字列長個であるという解の制約が必要である．ここでは，解きたい問題は $\mathcal{R}$ 個の出力要素で構成されているし，出力ベクトルの一つの要素のインデックスを $r$ で表す．このとき，この問題全体の解空間をベクトルの集合 $\mathcal{Y}$ で表す．また，入力が与えられた時に得られる一つの出力ベクトルを $\mathbf{z} \in \mathcal{Y}$ で表す．

次に，DD-ADMM の特徴として，出力の要素で任意のグルーピングを行う．この時の条件として，全ての出力要素は必ず複数のグループに属するように設計する．これは，DD-ADMM はいわゆる多数決をベースとして最終的な出力が決定されるため，一つの出力要素は 2 つ以上のグループに属してそれぞれ別の観点からの評価される必要があるからである．

ここでは，グループを $S$ 個作成したとする．$s$ 番目のグループに含まれる出力要素の部分解空間を $\mathcal{Y}_s$ とする．つまり $\mathcal{Y}_s \subseteq \mathcal{Y}$ である．このとき，あるグループ $s$ に含まれる $r$ 番目の極小部分問題の解を $z_s(r)$ とする．

$s$ 番目のグループで得られた推定値に対して，出力の良さを計算する関数を $f_s$ とする．ここでは，$f_s(\mathbf{z}_s)$ の値が大きいほど，出力として良いものであるという仮定のもとに計算される値だとする．

このとき，問題全体は以下の最適化問題として定式化できる．

$$\begin{aligned} \mathbf{z}^* = & \mathop{\arg\max}_{\{\mathbf{z}_s\}_{s\in\{1,\ldots,S\}}} \sum_{s=1} f_s(\mathbf{z}_s) \\ \text{s.t.} \quad & z_s(r) = u(r) \quad \forall s, r \in \bar{\mathcal{R}}_s \end{aligned} \tag{2}$$

ここで，$\mathbf{z}_s$ はグループ $s$ に含まれる極小問題の解の集合とする．また，$\mathbf{z}$ は，$z_s(r)$ の全集合とする．

この最適化問題の解は，$S$ 個のグループが独立に，各グループが担当する従属最適化問題に従って出力 $\mathbf{z}_s(r)$ を決定する処理と，同じ位置の変数は，各グループ間で一致しなくてはいけないという制約を満たすように変数の値を更新する処理の反復計算により求められる．

実際の導出式や最適化の方法は，文献 [1, 2] に詳細が記述されているので本稿ではスペースの都合上省略する．本稿での議論に重要な点は，式 (2) で示した実際に解きたい主問題の解を，細かく分割した従属最適化問題の集合を解くことで得られるということである．また，個々の従属最適化問題はお互いに独立であるため，並列処理なども容易に行うことが可能である．よって，DD-ADMM に基づいて問題を定式化する際には，従属最適化問題をうまく設定することに全てがかかっている．

本稿では，以下，本稿で対象とする問題に対して効率的なグループとその従属最適化問題の効率的な解法を示す．

## 4 拡張ラグランジュ緩和に基づく定式化

日本語の単語，文節，文分割および品詞付与問題を，前節で紹介した拡張ラグランジュ緩和に基づいて定義する．

図 1 で示したように，単語，文節，文分割の点推定モデルは，単純に，各文字間に分割するかしないかを表す一つの出力変数で表現できる．また，連結成分も接続する右と左の辞書エントリが出力として選択されたか，されなかったかの 0,1 で表現できるために，一つの変数を割り当てれば良い．最後に，辞書エントリに関しては，各位置毎に一つの出力変数を割り当てる．つまり例えば，長さが 3 の辞書エントリは 3 つの出力変数で表現される．従来は，mecab などにみられるように辞書エントリ一つに一つの変数を割り当てる方法が一般的だと思われるが，この部分が，今回の辞書のように可変長のエントリを効果的に扱うための提案法のトリックであり，若干変則的な部分である．

### 4.1 グループの設計

複数の問題を統合して得られた問題全体に対して，満たさなくてはいけない制約を記述するためにグループを作成する．以下に本稿で利用する 5 種類のグループタイプを述べる．

**グループタイプ 1(辞書エントリ内の整合性)**

前述のように個々の辞書エントリは，エントリ長に応じた数の出力変数で構成される．ある辞書エントリが，出力として選ばれる場合というのは，辞書エントリに含まれる全ての出力変数が 1 の場合である．逆に，出力として選ばれない場合とは，全ての出力変数が 0 の時である．

この条件に従うと，一つの辞書エントリに含まれる変数のいくつかが 1 で，いくつかが 0 といったようになると，そのエントリが出力に選択されるのかどうか判定できなくなってしまう．そこで，一つの辞書エントリ内の変数は必ず全て同じになるという制約を設ける．

この制約を表現するために，グループタイプ 1 では，各辞書エントリで一つのグループを生成する．よって，グループの総数は，ある入力に対して適用できる辞書エントリ数が $n$ だとすると，$n$ 個のグループが生成される．

この制約に対応する従属最適化問題は以下のようになる．

$$\begin{aligned} \mathop{\text{maximize}}_{\{z_s(r)\}_{r\in\mathcal{R}_s}} \; & -\frac{\rho}{2} \sum_{r\in\bar{\mathcal{R}}_s} \left(z_s(r) - a_s(r)\right)^2 \\ \text{s.t. } & z_s(r) = b \quad \forall r \in \bar{\mathcal{R}}_s \end{aligned} \tag{3}$$

制約 $z_s(r) = b$ が全ての変数が同じという制約を表している． 制約に従って $z_s(r) = b$ で置き換えると，以下の解析解が得られる (導出式は省略).

$$b = \frac{\sum_{r\in\bar{\mathcal{R}}_s} a_s(r)}{|\bar{\mathcal{R}}_s|} = z_s(r) \quad \forall r \in \bar{\mathcal{R}}_s \tag{4}$$

この式の意味は非常に単純で，グループ内の $a_s(r)$ の平均

が $z_s(r)$ になるということである．よって，グループタイプ 1 の従属最適化問題は $a_s(r)$ の平均を求める処理により解が得られる．

**グループタイプ 2(各文字位置毎の制約)**

各文字位置で出力として選択できる辞書エントリは当然必ず一つである．この制約を表現するために，グループタイプ 2 では，同じ文字位置の辞書エントリで一つのグループを生成する．よって，グループの総数は入力文字列長が $n$ だとすると，必ず $n$ 個のグループが生成される．

この制約は，各文字位置で 1 を取れる出力変数は一つで，それ以外は全て 0 になる．という制約式で表現できる．また，各文字列位置での出力変数の合計は 1 になる，という等価な制約でも表現できる．よって，グループタイプ 2 に対する従属最適化問題は以下の制約付き最適化問題で記述できる．

$$\begin{aligned} \underset{\{z_s(r)\}_{r\in\mathcal{R}_s}}{\text{maximize}} \ & -\frac{\rho}{2}\sum_{r\in\overline{\mathcal{R}}_s}\left(z_s(r)-a_s(r)\right)^2 \\ \text{s.t.} \ & \sum_{r\in\overline{\mathcal{R}}_s} z_s(r) = 1 \end{aligned} \tag{5}$$

これは従属最適化問題は，文献 [1, 2] で ‘XOR 問題’ として説明されている制約である．また，これは，$n$ 変数を $n$ 次元ベクトルで表現した際に，$n$ 次元多面体上への最短ユークリッド距離による写像処理と等価な処理としても知られている．

実際にこの写像処理は，出力変数のソートによって解を得ることができることが知られている [5]. よって，グループ内の変数の数を $n$ とすると，この制約の計算量は $O(n\log n)$ である．

**グループタイプ 3(辞書エントリと連結成分の制約)**

局所的な制約として，もしある辞書エントリが出力として選択されるのであれば，その辞書エントリへの連結成分も必ず一つ選択されていなくてはいけない．そうでないと，辞書エントリからの出力の整合性が取れなくなる．また逆に，もしその辞書エントリが出力として選択されないのであれば，その辞書エントリに接続している連結成分も必ず全て選ばれない状態でなくてはならない．

この局所的な制約を表現するために，グループタイプ 3 では，辞書エントリと右端または左端と，そこへ接続されている連結成分で一つのグループを構成する．つまりグループタイプ 4 の各グループには必ず一つの辞書エントリの出力変数と，一つ以上の連結成の出力変数で構成される．よって，グループの総数はエントリ数が $k$ 個だとすると，$2k+2$ 個のグループが作成される．ただし，最後の +2 部分は，入力の先頭と末尾を表す特殊エントリ，一般的に BOS, EOS などと記述されるエントリに接続される連結成分のグループである．

この制約条件は，辞書エントリに対応する出力変数が 1 ならば，連結成分の出力変数のうちどれか一つが 1 にならなくてはいけない，あるいは逆に，連結成分のうちどれか一つが 1 となるためには，辞書エントリの出力変数も 1 にならなくてはいけない，ということを意味する．或は，もし，辞書エントリに対応する変数が 0 なら，連結成分の変数はすべて 0 になる．

これは，文献 [1, 2] で紹介されている ‘XOR-wiht-OUT’ 形式の制約である．辞書エントリの変数を反転することで，グループタイプ 2 の XOR と同じアルゴリズムで解くことができる．よって，グループタイプ 3 も，グループ内の変数の数を $n$ とすると，$O(n\log n)$ の計算量で解くことができる．

**グループタイプ 4(連結成分と分割点推定の整合性)**

次に，分割点推定と連結成分の選択結果の整合性を考える．例えば，単語分割の場合，単語の分割点推定で，ある文字と文字の間で分割されると判定されるとすると，その文字位置の連結成分の必ず一つ選択されていなくてはいけない．これは，連結成分が選択されるということは，必ず辞書エントリの区切れであることを意味するので，必ず単語境界となるからである．逆に，ある文字位置で連結成分が選択されないということは，その文字位置をまたぐ辞書エントリが選択されてることを意味する．

そこで，グループタイプ 4 では，各文字位置での単語分割点推定と連結成分で一つのグループを構成する．文字列長が $n$ のとき，グループタイプ 4 は $n+1$ 個生成される．実際には，入力の先頭と末尾は，分割点は存在しないが，便宜上作成することとする．

この制約も，グループタイプ 3 と同様に，XOR-with-OUT 形式の問題である．よって，分割点推定の変数を反転することで，グループタイプ 2 の XOR と同じ処理で解を求めることができる．このグループに対しても，グループ内の変数の数を $n$ とすると，この制約の計算量は $O(n\log n)$ である．

**グループタイプ 5(分割点推定間の整合性)**

本稿では，単語，文節，文と複数の違った観点での分割点推定を同時に処理しようとしている．これら 3 つの処理の関係は単純に，文節境界は必ず単語境界の位置のみで起こり，同様に，文境界は，文節境界の位置でのみ起こるという，一種の包含関係にある．この制約を表現するために，同一文字位置の分割点推定同士で一つのグループを構成する．簡単のため，単語と文節，文節と文，単語と文といったように 2 つの分割点間でグループを構成する．よって，グループタイプ 5 では，文字列長が $n$ とすると，$3n$ グループが生成される．また，グループを構成する変数は必ず 2 に固定される．

2 つの分割点間の関係は，例えば，単語と文節の分割点とすると，変数がそれぞれ，$(1,0),(1,1),(0,0)$ の時には制約が満たされているが，$(0,1)$ の時には制約が満たされていない．と記述できる．

このとき，XOR のように，多面体上への写像処理としてこの制約を考える．問題を簡単にするため，より分割点が多くなる方の変数を反転してみる．すると，$(0,0),(0,1),(1,0)$ の時には制約が満たされ，$(1,1)$ の時には制約が満たされていないというように条件を変えることができる．こうすることで，2 次元空間で $(0,0)$ と $(0,1)$ と $(1,0)$ の 3 点で張られる三角形へのユークリッド距離最小化での写像問題として表現することができる．

この問題は簡単に解けて，まず，$\max(0,a_1)+\max(0,a_2)\leq 1$ の時は，XOR と同じアルゴリズムで解くことができる．また，それ以外の時も，$z_1=\max(0,a_1)$, $z_2=\max(0,a_2)$ という簡単な計算で解が得られる．

### 4.2 解析処理

以上の 5 つのグループタイプを使い，与えられた入力に対して，対応するグループを全て作成する．つぎに，DD-ADMM の最適化式に従って反復計算により出力変数を求める．収束したら，得られた出力変数のうち 1 となっているものを選択する．ただし，実際の最適化では，LP 緩和によって出力変数は 0 から 1 の間の値をとるため，かならずしも 1 であるとは限らない．この場合は，まず，辞書エントリは各文字位置で最も大きな値をとったエントリを入力の先頭からグリーディーに選択することとする．また，その辞書エントリの区切れに従って単語，文節，文境界を，各境界点推定の出力変数から獲得する．

以上の処理により，提案法は，単語，文節，文および品詞付与の同時モデルを処理する．

## 5 実験設定

本稿では，日本語の単語，文節，文分割および品詞付与の 4 つの自然言語解析問題を用いて提案法の有効性を検証する．これら 4 つの問題を従来のパイプライン形式で処理する場合と，提案法である同時モデルで処理する場合で解析精度を比較する．

### 5.1 データ

実験用のデータとして，新聞記事, blog, web 一般の 3 種類のデータを準備した．表 1 に実験に用いたデータとそ

表 1: 実験で使用したデータの文・単語数

| データ種類 | 学習用データ<br>文数 / 単語数 | 開発用データ<br>文数 / 単語数 | 評価用データ<br>文数 / 単語数 |
|---|---|---|---|
| 新聞記事 | 7,633 / 195,466 | 1,493 / 41,413 | 1,220 / 31,292 |
| blog | 27,855 / 391,664 | 4,266 / 54,336 | 4,723 / 56,738 |
| web 一般 | 25,613 / 465,186 | 1,966 / 39,952 | 2,696 / 58,404 |

の量を示す．新聞記事とは，京大コーパス 4.0 である[*1]．blog と web 一般データは，web から半自動的に生成したデータであり，人手により単語，文節，文，品詞情報が付与された独自のデータである．品詞情報は，naist-jdic に準拠したタグセットが用いられている．

品詞付与に用いる辞書エントリには，京大コーパスに対する実験に関しては，mecab 用に配布されている jumandic[*2]，それ以外のデータには，mecab 用に配布されている naist-jdic[*3]を再利用した．

### 5.2 モデル学習，比較手法

単語，文節，文分割のモデルは，それぞれ独立に online passive-agressive 法 [6] を用いて学習を行った．辞書を用いた品詞付与のモデルは，mecab を用いて単語エントリのコストと連接コストを計算し，そのコストを利用した．

提案法の有効性検証の方法として，従来からあるパイプライン形式の処理による解析精度と，分割モデル単体を独立に使用した際の解析精度を示す．このとき，パイプライン処理では，単語分割，品詞付与には mecab の実装[*4]をそのまま利用し，文節や文分割には，前述の online PA により生成した分割モデルを利用する．ただし，文節や文区切りは，その前の処理で決定された境界でのみ行われることとする．これによって，各境界整合性は保たれた結果が得られる．

一方，独立処理では，前の段階の解析を全く無視して解析を行った．つまり，分割点での出力の整合性は全く考慮しない場合の解析精度を意味し，解析結果として各境界の整合性がとれている保証はない．

### 5.3 結果および考察

表 2 に，実験結果を示す．表中の単語，文節，文分割の列の値は，セグメント F 値である．品詞付与の列は，品詞ラベル正解率であり，左列が品詞大分類の正解率，右列が品詞細分類の正解率である．ただし，品詞の正解率は，単語分割に誤りがある場合は必然的に誤りになるため，実質は単語分割と品詞付与の双方を合わせた正解率を意味している．

辞書エントリだけを用いて，単語，文節，文分割モデルを利用しない場合の解析精度 (辞書だけモデル) と，逆に，辞書を用いずに，単語，文節，分分割だけの同時モデル (分割同時モデル) との解析精度を比較すると，提案法を完全な形で用いることで，おおよそ全ての解析精度が向上していることがわかる．

一点，blog データの単語分割に関しては，辞書と分割モデルの同時モデルよりも，分割モデルのみの同時モデルの方が良い結果が得られた．これは，blog データの品詞推定は，そもそも曖昧性の高い難し問題であり，品詞を正しく推定しようとする結果，単語分割も間違える方向に引きずられて分割モデルのみの時よりも悪くなってしまったと考えられる．ただし，辞書情報を使うことで文節や分の分割精度は上がっているという興味深い結果となった．この結果は，むしろデータの性質を示していると考えられる．つまり，単語分割のみは比較的可能であるが，品詞は曖昧性が高くて正確に決定することは難しい．しかし，文節や，文といった区切りを決定する際には，品詞情報は有益であるといった性質があると思われる．

表 2: 実験の結果

(a) 新聞記事データ (京大コーパス) での実験結果

| | 単語<br>seg F. | 品詞<br>Acc. | 文節<br>seg F. | 文<br>seg F. |
|---|---|---|---|---|
| パイプライン | 97.26 | 95.55 / 94.41 | 94.68 | 92.56 |
| 分割点推定 (独立処理) | 93.21 | - | 94.41 | 92.22 |
| 提案法 | 97.67 | 95.88 / 94.77 | 94.98 | 92.61 |
| (辞書だけモデル) | 95.40 | 94.09 / 92.69 | - | - |
| (分割同時モデル) | 93.44 | - | 94.43 | 92.61 |

(b) blog データでの実験結果

| | 単語 | 品詞 | 文節 | 文 |
|---|---|---|---|---|
| パイプライン | 85.37 | 72.87 / 68.46 | 79.01 | 75.55 |
| 分割点推定 (独立処理) | 89.81 | - | 79.76 | 75.65 |
| 提案法 | 87.21 | 74.58 / 70.02 | 79.66 | 77.72 |
| (辞書だけモデル) | 84.97 | 72.45 / 68.16 | - | - |
| (分割同時モデル) | 90.25 | - | 79.60 | 76.95 |

(b) web 一般データでの実験結果

| | 単語 | 品詞 | 文節 | 文 |
|---|---|---|---|---|
| パイプライン | 95.56 | 94.00 / 91.85 | 75.20 | 77.12 |
| 分割点推定 (独立処理) | 92.09 | - | 76.88 | 77.55 |
| 提案法 | 95.97 | 94.20 / 92.03 | 76.86 | 77.44 |
| (辞書だけモデル) | 95.54 | 93.92 / 91.86 | - | - |
| (分割同時モデル) | 92.09 | - | 75.63 | 77.33 |

次に，従来のパイプライン処理との比較を行うと，明らかに提案法が高い遺跡結果が得られることが示されている．提案法は，様々な観点のモデルを統合して解析を行っているため，より広い観点を使って解析を行うことができているためだと思われる．

## 6 結論

本稿では，文献 [1, 2] で提案された DD-ADMM に基づいて，単語，文節，文分割，品詞付与問題を統合して同時に解析する方法を提案した．また，それらを効率的に処理するためのグループの作成法と，グループ内で定義される従属最適化問題の効率的な解法を示した．実験では，提案法である同時解析モデルは，従来のパイプライン方式の処理と比べて解析精度が向上することを確認した．これは，提案法の同時解析では，個々のモデルが持つ情報をお互いに効率的に補完することができていることを示していると考えられる．

## 参考文献


[1] Andre Martins, Noah Smith, Mario Figueiredo, and Pedro Aguiar. Dual decomposition with many overlapping components. In *Proc. of EMNLP*, pages 238–249. Association for Computational Linguistics, July 2011.

[2] André L. Martins, Mário A. T. Figueiredo, Pedro M. Q. Aguiar, Noah A. Smith, and Eric P. Xing. An augmented lagrangian approach to constrained map inference. In *ICML*, pages 169–176, 2011.

[3] Graham Neubig, Taro Watanabe, Eiichiro Sumita, Shinsuke Mori, and Tatsuya Kawahara. An unsupervised model for joint phrase alignment and extraction. In *Proc.of HLT-ACL*, pages 632–641, Portland, Oregon, USA, June 2011.

[4] Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. Applying conditional random fields to japanese morphological analysis. In *Proceedings of EMNLP 2004*, pages 230–237. Association for Computational Linguistics, July 2004.

[5] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient Projections onto the L1-ball for Learning in High Dimensions. In *proc. of ICML-2008*, pages 272–279, 2008.

[6] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online Passive-Aggressive Algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.


[*1] http://nlp.ist.i.kyoto-u.ac.jp/index.php?京都大学テキストコーパス

[*2] http://code.google.com/p/mecab/downloads/detail?name=mecab-jumandic-5.1-20070304.tar.gz

[*3] http://sourceforge.jp/projects/naist-jdic/

[*4] http://mecab.sourceforge.net/