情報アクセスシンポジウム2011

【レギュラー論文】

Web 文書を対象とする 情報抽出型検索エンジンのプラットホーム

菅 原 晃 平[†1]　森　辰 則[†2]

本稿では，研究成果の還元と応用研究の促進をするために，Web 上の文書に対して様々な情報抽出プログラム部品を組み合わせて利用できる情報検索・情報抽出を融合する統一的なプラットホームを提案する．特に開発者が手軽に自由度の高い部品を作れるような情報抽出のスキームを提案する．単純なパターンマッチングの例として発売日を抽出するシステムと提案システム上で利用可能な部品を作成し比較することで，開発者の労力が低減されていることと，部品を手軽に利用可能にすることの利点を確認した．

Search Engine type platform for information extraction using Web documents

Kouhei Sugawara[†1] and Tatsunori Mori[†2]

In this paper, we propose a platform of Search Engine which can be used a variety of program parts of information extraction for Web documents. It will be expected to contribute to returning useful results of various research to general users and promoting applied research. In particular, we propose a scheme of information extraction in which developers can program parts easily and freely. As a relativiely simple case study, we performed an experiment of developing a program extracting release dates. We programmed Release Date Extraction System and a program using pattern match, and compared them. We confirmed reducing developers' costs and increasing benefits when they used a proposal system.

†1 横浜国立大学大学院環境情報学府
Graduate School of Environment and Information Sciences,Yokohama National University
†2 横浜国立大学大学院環境情報研究院
Graduate School of Environment and Information Sciences,Yokohama National University

1. 序　　論

近年，計算機の高性能化やネットワークの発達に伴い電子化された文書が膨大な量となり，それら文書群から利用者が必要な情報を効率良く取得する為の情報アクセス技術が必須となっている．特に Web 上の文書にアクセスするために検索エンジンが広く使われている．

従来の検索エンジンはキーワードの入力により，関連した文書群のリストとそのスニペットを表示するものが一般的である．しかし，実際に利用者が必要な情報はその文書中の一部分にのみ存在することが多いために，利用者は検索エンジンの出力結果からリンクを辿り文書を端から端まで読まなければならないという労力をかけざるを得なかった．

そのための研究として様々な情報抽出の研究[1)]が行われている．我々はそれら研究成果を手軽に利用可能な情報抽出プログラム部品（以下，部品と呼ぶ）として Web 上に公開したいと考えている．何故なら，その部品の部品開発者（以下，開発者と呼ぶ．）と部品利用者（以下，利用者と呼ぶ．）はそれぞれ次のような問題を抱えているからである．まず，開発者はそのような部品を公開するためのシステム開発と保守の労力が大きな問題である．また，開発者は既存の部品を利用することで新しい部品を作りたい場合もあるが，統一的な情報抽出のスキームがないために，他人の実装を活用できず，本来開発したかった部分以外での再実装の手間が非常に大きかった．一方で，利用者は部品を手軽に利用することができないためにその恩恵を十分に享受できていない．さらに，提案された多くの情報抽出システムの中に含まれる部品は連携を前提にしていないために，利用者も様々な部品を組み合わせて利用することで簡単により複雑な処理を行うことができかった．

そこで本研究では様々な部品が共存し再利用可能な情報検索・情報抽出を融合する統一的なプラットホームを提案する．提案システムでは統一的な情報抽出のスキームとして，入力として文書のストリームを受け，その中の情報抽出箇所に XML 流のタグを付与するという開発者に馴染み深いモデルを採用した．部品はこのスキームに従って作成されることが期待される．さらに，前段の部品が付けたタグの情報を後段の部品が取捨選択して必要なタグのみを残し入力とすることで，部品における情報抽出の処理が簡潔になると共に，複数の部品を容易に組み合わせることが可能となる．利用者が文書群を絞り込むキーワードと共に，部品列を入力すると，提案システムでは部品を動作させ文書群にタグを付け，最後に抽出文字列のリストを出力する．提案システムの利用者は情報要求が関連文書ではなく，文書の一部分であるという想定であり，そうであれば従来の検索エンジンの結果に比べてさらに細かな単位で情報の絞込みを行うことができる．例えば，利用者がある文書群から価格情報を必要とする場合，利用者はその文書群を指定する入力

と情報抽出を行う部品列を入力する．この例では二つの部品からなる部品列が考えられ，前段の部品が文書中の数値を示す文字列に数値タグを付け，そのタグ付けられた文書を後段の部品の入力とする．後段の部品はタグの情報を利用し，「⟨数値⟩[⋆1]」を含む表現の内，例えば「⟨数値⟩円」の様に価格である文字列だけにさらにタグ付けを行う．最後に，後段の部品のタグ付けの結果を利用して価格情報の抽出結果リストとして利用者に提示する．

この様にすることで，開発者は部品の開発に注力し，システム開発と保守から解放されると共に，さらに他の開発者が作った部品を利用した新たな部品を作り出すこともできる．また，利用者も様々な部品を自由に組み合わせて利用できるため，多様な情報抽出を行うことができる．我々の目標は様々な質の良い部品が利用可能となり，利用者がそれらの部品を利用することで，利用者が必要な情報を効率よく取得できるような情報抽出型検索エンジンをサービスとして提供することである．

2. 情報抽出システムに対する要求の検討

本章では，まず情報抽出システムの要求について開発者と利用者の二つの立場から検討する．次に，その要求を満たす情報抽出のスキームを検討する．さらに，その提案する情報抽出のスキームで必要なそれぞれの処理について検討する．

2.1 開発者と利用者の要求

本節では様々な情報抽出システムに対する要求を満たすためには一般的に何ができる必要があるのかを開発者と利用者の二つの立場で検討を行う．まずは，開発者の立場からすると部品を手軽に作ることができることが重要である．そのためには部品の入出力のモデルをできる限り簡潔にすることが必要である．また，本稿ではこれ以上触れないが，バージョン管理が容易であったり，既存の有用な部品を簡単に検索し利用できる事も重要である．また，情報抽出の研究では一般的に「抽出する情報の型」が決められる．例えばある製品の情報を抽出したい場合に「製品名」，「値段」，「発売日」などが抽出対象の型として定義される．そのため，開発者がこのような型を自由に定義できる必要がある．さらに，情報抽出においてパターンマッチングの技術がよく用いられる．パターンマッチングとは「⟨数字⟩日」といったパターンを予め定義しておき，それを「日付」であるとして情報抽出を行う手法である．この様な場合，「数字」を抽出する部品の抽出結果を利用して「日付」を抽出する部品を作れることが望ましい．情報抽出ではこの様な段階的な絞込みによる抽出がよく用いられるため，複数の部品が組み合わせ可能であることと，後段の部品が前段の部品の抽出結果を自由に参照できることが必要である．

一方で，利用者の立場からすると要求にあった部品が簡単に利用できることが重要である．また，利用者の多様な情報要求を満たすためには個別の状況に細かく対応できる情報抽出や，複数の部品を自由に組み合わせて利用できることが必要である．さらに，部品の入出力の仕様を利用者が理解しやすくすることで，利用者の意図しない結果が返ってきた場合にもシステムに対する入力を修正できるようになる．

2.2 情報抽出のスキーム

本節では開発者が部品の作成を行うための情報抽出のスキーム，抽出結果の表現方法について検討を行う．部品同士が連携を行うためには，どのような手段で情報を伝達するかが重要である．言語データを扱う統一した形式として XML 流のマーク (タグ) が使われるのが一般的となっている[2)]．そこで，我々は開発者に馴染み深い XML 流のタグを用いた情報抽出のスキームを提案する．開発者はタグ名を用いて「抽出する情報の型」を表現し，抽出箇所の文字列を「⟨タグ名⟩抽出文字列⟨/ タグ名⟩」の様にタグで囲った文書を出力する部品をプログラムとして作成する．この様にタグを用いて抽出結果を表現することで，部品の入力は文書であり，その出力は情報抽出箇所にタグが付いた入力の文書という様にモデルを単純化することができる．また，その出力された文書をさらに別の部品の入力とすることで，文書のストリームのみで情報を伝達できると共にパイプライン型の処理となるため，自由に部品を組み合わせることも可能となる．しかし，複数の部品が自由にタグを付けた場合に，そのタグが付いた文書を渡される部品の開発においてはタグの入れ子関係や，必要のないタグを考慮しながらプログラミングを行わなければならない．これは非常に煩雑であり，前段の部品やその組み合わせに依存してしまう．そのため，入力において部品が必要なタグを指定し自由に取捨選択できる必要がある．

2.3 構造化のためのタグの必要性

2.2節で述べたタグ付けの方法では，文書中に存在する任意の文字列を抽出結果として表現することができる．しかし，抽出する情報の型によっては，単に文字列を抽出するだけでなく，複数個所の抽出結果をまとめて一つのデータとして扱える必要がある．

例えば，ある部品が「Yokohama National University」から「YNU」を抽出する場合に，「⟨略称⟩Y⟨/ 略称⟩okohama ⟨略称⟩N⟨/ 略称⟩ational ⟨略称⟩U⟨/ 略称⟩niversity」という様なタグを付けることになる．この場合，それぞれのタグで囲われた「Y」と「N」と「U」を連結して「YNU」であるというを表現できる必要がある．また，「⟨駅名⟩新横浜⟨/ 駅名⟩の次は⟨駅名⟩品川⟨/ 駅名⟩，⟨駅名⟩東京⟨/ 駅名⟩」の様なタグが付けられた文がある場合に，路線情報として抽出するならば，「新横浜→品川→東京」の様な順序関係を表現できる必要がある．この様

⋆1 簡単のために⟨数値⟩...⟨/ 数値⟩に該当する文字列を単に⟨数値⟩と書くことがある．

な要素の型が等しい順序付きデータ構造をリスト (List) と呼ぶ．また，地域毎に駅名を抽出するならば，「新横浜」は神奈川県に属し，「品川」や「東京」は東京都に属すといった共通点を持つ細かなまとまりを表現できる必要がある．この様な要素の型が等しく順序に興味がないデータ構造を集合 (Set) と呼ぶ．さらに，「⟨製品名⟩ABC⟨/ 製品名⟩ は ⟨値段⟩1000 円 ⟨/ 値段⟩ で発売された．」の様なタグが付けられた文がある場合に，製品情報として抽出するならば，製品「ABC」の値段が「1000 円」であるといったことを表現できる必要がある．この様な要素が複数の型を持ったデータ構造を組 (Tuple) と呼ぶ．より高度な情報抽出を行うためにも，これらデータ構造をタグによって表現できる必要がある．

2.4 部品に渡される文書の内容

本節では部品の入力として渡される文書について含めるべき内容について述べる． Web 上の一般的な文書として HTML 文書を対象にして情報抽出をする際には,HTML タグを削除した後に処理を行うことが多い．しかし, 情報抽出をする際に TABLE など HTML タグの情報や構造を利用することや，リンク先など HTML タグ自身が抽出対象になることもある．そのため，部品に渡される文書は HTML タグを除いた文と HTML タグを含めた全文を選択できることが必要である．

2.5 組み合わせた部品の有効活用

本節では複数の部品の組み合わせをどのように扱うべきであるかを述べる．提案する情報抽出のスキームはパイプライン型の処理であるために，出力結果は複数の部品の組み合わせか単独の部品を使用したのかを区別をする必要がない．また，開発者や利用者が良い部品の組み合わせを発見した場合には，その組み合わせを共有し他の開発者や利用者が再利用できることが必要である．

さらに，部品の連携において，提案する情報抽出のスキームでは，「抽出する情報の型」，つまりタグ名とその抽出文字列で情報が伝達される．そのため，部品の組み合わせをより柔軟にするためには，部品が付けるタグ名を容易に変更できる必要がある．

2.6 利用者の入力とシステムの出力

本節では情報抽出システムの利用者に求めるべき入力とその出力について述べる．情報抽出の処理時間を現実的にするためには，より処理時間が短い情報検索で情報の絞り込みを行った上で必要な部品だけを動作させ情報抽出を行うことが望ましい．そのため，利用者の入力には絞り込みを行うための入力と必要な部品の選択が必要となる．

また，利用者に提示する結果には，部品によってタグ付けされた文書群だけでは情報が煩雑である．そのため，文書毎にタグ付けされた「抽出する情報の型」とその抽出文字列をリストにして簡潔に表示する必要がある．さらに，情報抽出結果を利用する際に利用者が元文書の情報抽出箇所にアクセスできる必要がある．何故なら情報抽出箇所にアクセスできなければ, 利用者は従来の検索エンジンの結果と同様に文書を端から端まで読まなければならないためである．

3. 提案システム

本章では 2 章で述べた要求を満たす情報抽出システムの一例として，提案するシステムについて述べる．まず，提案する情報抽出スキームを用いた提案システムの概要について述べる．次に要求を満たすための処理の実装について述べ，最後にその実装方法について述べる．

3.1 提案システムの概要

提案システムの流れを図 1に示す．提案システムでは部品を開発し，データベースに登録する開発者とその登録された部品を利用し情報抽出の要求をする利用者がいる．開発者は情報抽出箇所にタグを付ける部品を作成し， 3.5節で述べるように部品 DB のデータベースに部品を登録する．利用者は 3.6節で述べるように文書群を絞り込むキーワードと部品列の入力を行う．それに応じて提案システムは文書群の取得を行った後に部品を動作させ文書群にタグを付け，最後に部品がタグを付けた抽出文字列のリストを利用者に出力する．

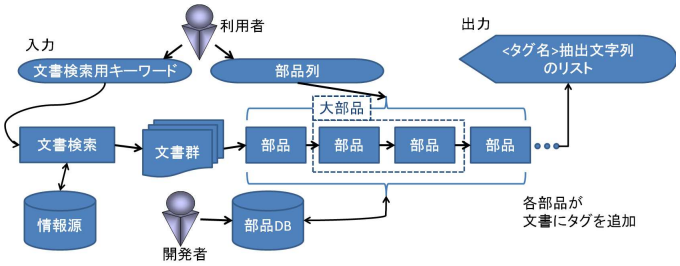

図 1 提案システムの処理の流れ
Fig. 1 Processing flow of the proposed system

3.2 タグの管理方法

本節では 2.2節で述べた情報抽出スキームの実装について述べる．提案システムでは Web 上の HTML 文書にタグを埋め込む上で既存の HTML パーサーなどの解析を阻害させないように「⟨!–$ タグ名 –⟩」の様なコメントタグでタグを表現することにする．また，部品が必要なタグを自由に取捨選択できるようにするために，部品が付与したタグの情報は文書とは独立に管理を

行い，部品の指定するタグを付与した文書を部品の入力としている．例えば，部品が数値タグを利用して情報抽出を行う場合，数値タグのみを指定すれば，入力される文書中には数値タグ以外の（他の部品が付けた）タグを含まれないことが保証される．

3.3 構造化のためのタグ

本節では2.3節で述べた構造化のためのタグの実装について述べる．それぞれの抽出文字列が関係を持ったデータ構造の情報をタグで表現するために, 属性を持ったタグ（関係タグと呼ぶ．）を定義し表現する．提案システムでは関係タグは「⟨!–% タグ名 属性 =”属性値 ”...–⟩」の様に表現する．提案システムでは List， Set， Tuple を表現するための属性を定義する．まず，同一の「抽出する情報の型」（タグ名）を用いた関係タグで複数のまとまりを扱うために，関係のまとまりを識別するための属性「コレクション ID （CID）」を定義する．その属性値は数値であり，数値が同じであれば同一のまとまりであると考える．また， List の順序を表現するための属性「順序（Num）」を定義する．その属性値は数値であり，その数値が順序を表わすものと考える．さらに， Tuple の要素の型を表現するために，タグ名のみを属性値とする属性「要素名（Name）」を定義する．関係タグの属性に Name が含まれておりその Name の属性値に重複がない場合は Tuple， Tuple ではなく Num が含まれていれば List，それ以外は Set であると考える．この様にすると， Tuple にも Num が存在する場合があるなど曖昧性が生じるが，より柔軟に関係を表現できるようにするためにこの様に定義する．例えば 2.3節で述べた「⟨!–$ 製品名 –⟩ABC⟨!–$/ 製品名 –⟩ は ⟨!–$ 値段 –⟩1000 円 ⟨!–$/ 値段 –⟩ で発売された．」という文を Tuple を用いて情報抽出すると，「⟨!–% 製品 CID=”1” Name=”製品名 ”–⟩⟨!–$ 製品名 –⟩ABC⟨!–$/ 製品名 –⟩⟨!–%/ 製品- ⟩ は ⟨!–% 製品 CID=”1” Name=”値段 ”–⟩⟨!–$ 値段 –⟩1000 円 ⟨!–$/ 値段 –⟩⟨!–%/ 製品 –⟩ で発売された．」の様にタグを付けてその関係を表現する．ただし，このままでは情報が煩雑であるので，後段の部品は製品タグのみを付与された文書を入力にすると考えられる． CID の属性値は提案システムが必要に応じて発行し，それ以外の属性値は部品が指定することにする．

3.4 文書の管理方法

本節では2.4節で述べた要求を満たす，部品の入力として渡される文書の処理内容について述べる．提案システムでは Web 上の文書に対して加工を行うため，その文書を一時的に保存する．また，部品が HTML タグを含めるかを選択するために，提案システムでは部品に HTML タグを含めて全文を取得する機能と HTML タグを除いた文を取得する機能の両方を提供している．部品が HTML タグを除いた文にタグを付けても提案システムで HTML タグを含めた全文での対応関係を計算し修正を行うため, 正しくタグ付けが行うことができる．

3.5 部品の登録

提案システムでは開発者は部品を利用可能にするために予めデータベースに登録する必要がある．部品をデータベースに登録することで利用者や開発者が部品名やタグ名，詳細説明の一部をクエリとして部品を検索し，部品の仕様を自由に参照した上で，部品名により部品を選択できるようになる．そこで, データベースに登録する部品の情報について詳しく述べる．

部品を登録する場合には, その部品名と部品がタグを付ける可能性のあるタグ名が必須事項である．さらに, 必要であれば部品が受け付ける引数（3.6節で後述）の省略可能性, その部品を用いる時に前段に必要とする部品やタグ名を登録することができる．また， 2.5節で述べた部品の組み合わせを共有するために，その組み合わせを大部品としてデータベースに登録可能にした．大部品を登録する場合にはその部品名と構成要素となる一つ以上の部品，並びにその部品に渡す引数を登録することが必須事項である．大部品にも部品名を与えることで，利用者は部品と大部品を区別なく部品名により部品を選択できる．さらに，構成要素の部品が付与するタグ名を大部品の出力の段階で別のタグ名に置き換えることと，大部品の出力するタグを構成要素の部品が付与するタグの中から指定することを可能にした．この様にすることで，ある部品が付与するタグ名を他の開発者が自由に変更し，組み合わせる部品が対応しているタグ名に置き換えるなど，より柔軟に部品を組み合わせることが可能となる．これらの登録情報は利用者に対する情報提供のためだけでなく，提案システム上で利用者の入力の妥当性の確認にも使われる．

その他にも利用者に提供される情報としてどのような抽出を行うかの詳細説明や必要な引数の詳細, 部品の入出力例を登録することが必要となる．これらの登録情報を参照することで，利用者は部品の入出力を理解しやすくなる．

3.6 利用者の入力と提案システムの出力

本節では2.6節で述べた要求を満たす，提案システムの入出力の仕様について述べる．まず，出力結果の例を図 2に示す．図 2は 4.2節の例の結果であり，利用者の入力のインターフェースと上位一件の文書のタイトル，並びにその抽出結果を示している．利用者は図中の１のテキストボックスに文書を絞り込むキーワードと２のテキストボックスに部品列を入力し，「検索」ボタンを押すことで情報抽出を行うことができる．出力結果は３の「抽出した情報の型」，つまりタグ名とその抽出文字列, その周り一文のスニペットが文書毎にリストとして表現される．また，その抽出文字列のリンクを辿ると４の情報抽出箇所にアクセスできる．

次に，提案システムで受け付ける入力の仕様について述べる．提案システムで受け付けるクエリの文法を EBNF 記法を用いて図 3の様に規定した．文書を絞り込むためのキーワードには図 3に規定される ⟨DocQuery⟩ を使用し，部品列には ⟨CompQuery⟩ を使用する．文書を絞り込

図 2 提案システムの入出力例
Fig. 2 A Input and Output example of the proposed system

むためのキーワードには， site コマンドを使用して指定したサイト内で文書検索することや，url コマンドを使用してキーワードを使用せずに URL で直接文書を指定することが可能である．また，部品列の入力では，複数の部品がある時には左から順に部品が実行され，前段の部品の出力（タグ付けされた文書のストリーム）が後段の部品の入力となる．また，データベース中に複数の同一の部品名がある場合には使用回数の多い部品が自動的に選択されるが，開発者名を付けることで部品を明示的に選択することができる．開発者名とは部品開発者を特定するための名前であり, 開発者が事前に登録する名前である．提案システムでは「開発者名. 部品名」で一意に部品が定まるようにしている．さらに，部品には任意の数の引数を図 3に規定される ⟨Arg⟩ に従って記述することで与えることができる．ただし，与えた引数が部品内部でどのように用いられるかは部品に依存する．複数の部品を用いた場合には複数のタグ名が存在する場合があるため, 利用者は出力結果に全てのタグ名について出力するか, 最後の部品が付けたタグ名について出力するかをチェックボックスで選択する．最後の部品が付けたタグ名にするのは, 利用者は本来タグ名を知る必要はなく，連携を前提とした複数の部品では最後の部品の結果が最も重要であると考え，タグ名を入力する必要がない選択方法を取ったためである．

従来の検索エンジンの入力とは違い, 利用者は登録済みの部品名を入力しなければならない．そこで, 提案システムではデータベースに登録された部品情報を利用し Ajax による部品名の入力補助を行ったり, 引数の数を満たさない場合や部品が存在しない場合はサーバーへのリクエストを行わずエラーを表示し待ち時間を短くしたりすることで利用者の負担を減らす努力を行って

```
<DocQuery>::=(<keyword>(<space>+<keyword>)*)|
((<keyword><space>+)*<UrlOpt>(<space>+<keyword>)* )
<UrlOpt>::=("url:"<url>("&"<url>)*)|("site:"<urld>("&"<urld>)*)
<CompQuery>::=<Comp>(<space>+<Comp>)*
<Comp>::=(<developer>".")?<compname>
(<lp><space>*<Arg>(<space>*<comma><space>*<Arg>)<space>*<rp>)?
<Arg>::=(<quote><argstr1><quote>)|<argstr2>
<keyword> は空白を除く任意の文字列
<space> は空白文字
<url> は任意のサイトの URL
<urld> は任意のサイトのドメイン名
<developer> は開発者名に成り得る任意の文字列
<compname> は部品名に成り得る任意の文字列
<lp> は左括弧
<rp> は右括弧
<quote> は引用符
<argstr1> は <quote> を除く任意の文字列
<argstr2> は <space><lp><rp><quote> を除く任意の文字列
```

図 3 入力のクエリ文法
Fig. 3 A query grammar of the input

いる．また，カテゴリー名や部品検索機能による視覚的に部品の入力を補助するインターフェースを追加することで正しい部品名の入力を簡単に行うことができる．ただし，カテゴリー名は部品を検索しやすくするための分類名であり，開発者が部品を登録する際に含める情報である．さらに開発者が登録した部品の詳細や入出力例を利用者が必要によって参照することで目的にあった正しい部品を選択できると考えている．

最後に，提案システムの出力の実装について述べる．部品間ではコメントタグで情報が伝達されるが, 利用者に部品が付けたタグの結果をそのまま表示してもコメントタグであるためブラウザ上には何も表示されない．そこで, 利用者に情報抽出箇所を表示する際にはタグで囲われたブラウザ上で表示できる文字列, つまり HTML タグを除いた文字列を色を付けた SPAN タグで囲い強調表示する．さらに瞬時にその文字列にアクセスできる様に親ノード又はその親ノードの直下に id の属性を埋め込むことで id によるページジャンプが可能となる．ただし, これらの元文書の強調表示やページジャンプを行うことは HTML の構造を改変する必要があるので JavaScript のクロスドメインの問題などから実際に元文書で行うことは難しい．そこで, 提案シ

ステムではキャッシュした文書にこれらの修正を加え, キャッシュされた文書であることを明示した内容を加え，動的にサーバー上でページを生成することで抽出文字列のリンクを辿ると擬似的に元文書の情報抽出箇所へアクセスすることができる．

3.7 実 装 方 法

本節では提案システムの実装方法と開発者が行う部品の実装方法について述べる．

3.7.1 提案システムの実装方法

提案システムは Java(Java Servlet) と HTML で書かれている．文書検索には Yahoo!API[3] を利用している．また，部品は Apach Derby[4] のリレーショナルデータベースに格納され，部品の探索性能はこれに依存する．さらに, 文書をキャッシュする際や結果を表示する際の HTML の構造の解析にはオープンソースである HTMLParser[5] を利用している．

また, 部品が付けたタグはキャッシュされた文書とは独立に XML 文書として管理される．そのため, 部品が参照する際などタグの探索性能はこの XML の構造に依存することになる．

3.7.2 部品の実装方法

部品の実装については開発者が行うことになる．そこで，部品の仕様の統一と開発者の負担の軽減を目的として，提案システムではデザインパターンの一つである Template Method パターン[6] を利用する．開発者は予め用意された抽象クラスのサブクラスとして Java のプログラムにより定義し，文書入力時のタグ付け処理のメソッドを実装する．提案システムではそのサブクラスのインスタンスを生成し，該当メソッドを呼び出すことで部品を使用している．また，セキュリティの関係上，部品クラスにファイルの読み込み・書き込み権限を付与せず，提案システムで提供されたクラスを用いて部品は文書の読み込みやタグ付けを行う．その際に部品が求めるタグや文書の形式を指定する．タグ付けの際は開始タグ，終了タグ共に部品が文書の先頭を 0 としたオフセット値と必要があればタグ名や関係タグの属性値を指定し，該当メソッドを呼び出す．

4. 想定される部品の例・その利用例

本章では提案システムにどの様な部品が登録され, どの様に利用者に利用されるかという例を利用者の視点から挙げる．

4.1 最も単純な抽出例

最も単純な例として, 引数なしの部品を利用する場合を挙げる．例えば, **価格**[⋆1] という部品が登録されたとし, **価格**は「⟨数値⟩ 円」や「¥⟨数値⟩」の表記を抽出する部品とする．例えば, 利用者はある製品 ABC の価格を知りたい場合に文書を絞り込むためのキーワードに「ABC」[⋆2] , 部品列に「**価格**」を入力する．その様にして, 利用者は必要に応じて抽出文字列の周りの情報も確認し, 製品 ABC の価格を知ることができると考えられる．

4.2 引数に応じて抽出する例

引数を受け取る部品はツールの様に利用されることを想定する．例えば,**NP** という部品が登録されたとし,**NP** は第一引数に入力された文字列を含む名詞句を抽出する部品とする．そこで, 例えば利用者がノーベル化学賞を受賞した下村氏が発見した物質の名前を思い出せないとする．しかし, その物質の名前は「G」が含まれ「クラゲ」にある物質であるということを利用者が覚えている場合には, 文書を絞り込むためのキーワードに「ノーベル化学賞 クラゲ」, 部品列に「**NP(G)**」を入力する．そのようにしてその他の G を含む名詞と共に「GFP」が抽出され, 利用者はその物質の名前が GFP であることを思い出す．従来の検索エンジンでは「G」というクエリを含めても意味を持たない場合が多いが, 部品の引数に入力することで「G」という情報をクエリに含めることができる．この様に直感的な部品名ではないが, 部品を一つのツールとして利用者が提案システムを利用することができる．

4.3 文書分類のための利用例

少し極端な例として, 抽出された文字列が重要でない利用例を挙げる．例えば, **肯定否定判別**という部品が登録されたとし, **肯定否定判別**は文書に含まれる肯定的な語, 否定的な語の数を調べ肯定的な語が多ければ「肯定」というタグ名, 否定的な語が多ければ「否定」というタグ名で文書のタイトルにタグを付ける部品とする．そこで, 例えば利用者がある製品 ABC の評判を知りたい場合に, 文書を絞り込むためのキーワードに「ABC 感想」, 部品列に「**肯定否定判別**」を入力する．そのようにして, 利用者は各文書毎に肯定的であるか, 否定的であるかの分類結果を大域的に眺めることができるためその評判を知るとともにより興味のある文書を即座に閲覧することができると考えられる．抽出文字列に肯定, 否定といった語が出ない場合でもタグ名を予め定義しておくことで文書に出現しない語を利用者に伝えることができる．

5. 評 価 実 験

提案システムにより開発者の労力を低減できるかを検証するために評価実験を行った．河又ら[7] は業務アプリケーションの開発に専門的なメディア解析処理を意識せずに開発できるプラットフォーム IVCP を提案し，その評価として家庭の消費電力解析エンジンを使った業務アプリ

⋆1 部品は太字で示す.

⋆2 「ABC 価格」とした方が，価格を含んだ文書をより多く取得できると期待される.

ケーションの開発を行い，IVCP の利用前と利用後の開発量をコードライン数を用いて比較を行った．そこで，本研究でも部品開発者が作成するコードライン数を用いて労力が低減できているかを評価する．評価実験として単純なパターンマッチングを用いた情報抽出の部品を作成し，その単独システムと提案システムで複数の部品を組み合わせて作った大部品の開発者が作成するコードライン数とその機能分析を行った．比較結果を表 1に示す．ただし，コードライン数はコメントや空行を除いたプログラムのソースコードの行数とする．また，これらの値は提案システムのコードライン数ではない．

パターンマッチングの例として製品の発売日を抽出することを目的とし「⟨日付⟩ に発売」というパターンを作成し，このパターンにマッチした文字列を抽出する部品を作成する．単独システムでは製品名を利用者からの入力とし，文書検索エンジンに「(製品名) ”日に発売 ”」をクエリとし文書群の絞り込みを行い，正規表現を用いて「[0-9]+? 日に発売」にマッチする文字列を抽出する．単独システムでは入力にオプションは含めず，文書の保存や抽出箇所の記録を行わない．作成した単独システムの動作例を図 4に示す．なお，提案システムでの結果も同様の抽出結果となるため，図は省略する．

また，提案システム上で動作する大部品は正規表現の入力を受け付ける部品 **regex** と「⟨タグ名⟩ に発売」という様なタグ名を用いてパターンを指定した上でパターンマッチングを行える部品 **PatternMatch** を作成し，それぞれの部品のタグ名は部品名と等しくし，「**regex([0-9]+?日) □ PatternMatch(⟨regex⟩ に発売)**」の組み合わせを大部品**発売日**として登録した．大部品の出力するタグは ⟨PatternMatch⟩ のみに指定すると，この大部品は「⟨日付⟩ に発売」にマッチした文字列を抽出する部品となる．利用者は文書を絞り込むキーワードとして「（製品名）」[*1]，部品列の入力として「**発売日**」を入力する．

単独システムと大部品を比較すると，単独システムでは情報抽出箇所を記録する必要がなければ文書にタグを付ける必要がないために，文字の位置を考えず読み込んだ文字を加工してくことで情報抽出を行うことができるためアルゴリズムが単純になる場合がある．しかし，今回作成した単独システムは最低限の機能であるために，提案システムの様にシステムとして様々な機能を付加すれば開発者の情報抽出プログラム部品以外の開発量もさらに増加する．一方で，大部品では単独システム全体と比較した場合にはコードライン数を削減することができる．また，引数を受け付ける汎用性の高い部品を作成すれば提案システム上で手軽に利用できるため，ソースコードを書き換えることなく開発者や利用者が多様な情報抽出を行うことができる．例えば大部品**発売日**の場合，パターンマッチングの日付に相当する部分を他の表記も抽出できるような正規表現や部品に置き換えたり，「発売日は ⟨日付⟩」という様な新しいパターンを追加したり置き換えたりすることが提案システムでは可能である．

表 1 機能別コードライン数の比較結果
Table 1 Functional lines of code compared

機能	単独システム	大部品
入力処理	71	0
文書取得	545	0
パターンマッチング	69	89(regex38,PatternMatch51)
出力処理	59	0
計（行数）	744	89

⋆1 「（製品名） ”日に発売 ”」とした方が，マッチする文字列を含む文書が多くなると期待される．

図 4 単独システム動作例
Fig. 4 The output example of the stand alone system

6. 関 連 研 究

大島ら[8] は，Web 研究におけるソフトウェア試作時のコストを低減するためのライブラリ SlothLib 提案している．SlothLib は Web 検索 API サービス，WEB 上の情報取得と文字コード判別，既存のクラスタリングアルゴリズム，自然言語処理ツールなどの機能をまとめることで，Web 研究におけるソフトウェア試作を迅速にすることを目指している．これに対して，本研究ではその様な必要な機能を提案システム上で提供した上で，Web 研究の成果を提案システム上で部品として共有し，部品を再利用し組み合わせることでさらなる研究の発展を目指している．特に本稿では開発者が部品を作り易いように部品の組み合わせを容易にする情報抽出のスキームについて検討を行った．

関ら[9] は，効率良く検索を繰り返すための検索システムとして，半構造化文書群を対象と

して，その部分構造ごとに着目した多面的検索システムを開発した．多面的検索システムでは検索結果を指定した２つの部分構造に関するマトリクスとして表示することで，利用者が観点やより深い内容を再検索できるようにしている．Cafarella ら[10] は検索エンジンは人間の質問に応えるために設計されており自然言語処理アプリケーションをサポートすることに満足しないこと，そのアプリケーションのクエリによりサーチエンジンの負荷という問題より Binding Engine を提案している．Binding Engine は型変数と文字列処理の関数を含んだクエリをサポートし，独自のインデックスにより「powerful ⟨noun⟩」や「Cites such as ProperNoun(Head(⟨NounPhrase⟩))」の様なクエリについて素早く検索することができる．従来の応用的な検索エンジンは，一つのクエリに文書の絞込みから情報抽出の処理を含めた情報要求を入力するために，利用者の意図しない結果が返ってきた時に利用者はクエリを修正することが困難であった．しかし，本研究では入力の形式として従来の検索エンジンに入力する様なキーワードと使用する部品列という２つのクエリを新しく定義することで，文書の絞込みと情報抽出の処理を切り離し，利用者の意図した情報要求により正確に応えることができるようになると考えられる．

また，商業用のサービスに Google App Engine[11] がある．Google App Engine は Google, Inc. のインフラストラクチャー上で自作の Web アプリケーションを実行できるというサービスである．Google App Engine を利用すれば，Web アプリケーション開発者はサーバーの保守や負荷分散のための労力が解消される．本研究では，情報抽出に特化しシステムとして開発者の成果を共有することで，開発者は部品の開発に注力し，利用者が部品を自由に組み合わせて使用できる検索サービスを目指している．

7. 結　　論

本稿では，研究成果の還元と応用研究の促進するために，Web 上の文書に対して様々な情報抽出プログラム部品を組み合わせて利用できる情報検索・情報抽出を融合する統一的なプラットホームを提案した．単純なパターンマッチングの例として発売日を抽出するシステムと提案システム上で利用可能な部品を作成し比較することで，開発者の労力が低減されていることと，部品を手軽に利用可能にすることの利点を確認した．今後の課題として，既存の研究成果を部品化することや提案システムの外部公開，処理速度向上のために負荷分散の機能を実装することや，タグの情報管理の仕方を最適化することなどが挙げられる．また，本稿では触れなかったバージョン管理の仕方や部品の検索要件なども検討し，有用性を実証するために外部公開をして実際に提案システムを運用・評価・改善することが必要である．

謝辞　本稿を執筆するにあたり，ご討論，ご指導を戴いた横浜国立大学環境情報研究院渋木英潔博士に感謝する．なお，本研究の一部は，科学研究費補助金基盤 (C)No.22500124 によるものである．

参　考　文　献

1) Takahiro, F., Manabu, O. and Tsuneaki, K.: Trends in Text Processing Research : Role of Evaluation Workshop in Information Extranction, Automatic Text Summarization and Question Answering(⟨Special Issue⟩NTCIR from the View Point of Participant), Journal of Japanese Society for Artificial Intelligence, Vol.17, No.3, pp. 301–305 (online), available from ⟨http://ci.nii.ac.jp/naid/110002809016/en/⟩ (2002-05-01).
2) 長尾　真，佐藤理史，黒橋禎夫，角田達彦：自然言語処理，岩波書店 (1996).
3) Corporation., Y. J.: Yahoo! デベロッパーネットワーク - Yahoo! 検索（ウェブ検索）の検索パラメータ仕様，Yahoo Japan Corporation.（オンライン），入手先⟨http://developer.yahoo.co.jp/other/query$_p$arameters/search/websearch.html⟩（参照 2011-07-07）.
4) Foundation, A. S.: Apache Derby, Apache Software Foundation (online), available from ⟨http://db.apache.org/derby/⟩ (accessed 2011-07-07).
5) Geeknet, I.: HTML Parser, Geeknet, Inc. (online), available from ⟨http://htmlparser.sourceforge.net/⟩ (accessed 2011-07-07).
6) Gamma, E., Helm, R., Johnson, R. and Vlissides, J.M.: Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Professional (1994).
7) 河又恒久，有熊　威，白石展久，小山和也，奥村明俊：情報価値創造基盤の提案，人工知能学会全国大会 (2011).
8) 大島裕明，中村聡史，田中克己：SlothLib: Web サーチ研究のためのプログラミングライブラリ，日本データベース学会年次大会，Vol.6, pp. 113–116 (2007).
9) 関　隆宏，和多太樹，山田泰寛，廣川佐千男：検索支援と分析のための多面的検索システム，日本データベース学会年次大会 (2007).
10) Cafarella, M. J. and Etzioni, O.: A search engine for natural language applications, Proceedings of the 14th international conference on World Wide Web, WWW '05, New York, NY, USA, ACM, pp. 442–452 (online), DOI:http://doi.acm.org/10.1145/1060745.1060811 (2005).
11) Google, I.: Google App Engine - Google Code, Google, Inc. (online), available from ⟨http://code.google.com/intl/ja/appengine/⟩ (accessed 2011-07-07).