# 構造/テキストWebデータを対象とした Pay-as-You-Goスタイルの問合せ構築支援手法

安永　ゆい† 　森嶋　厚行††,††† 　袖山　広輝†

† 筑波大学大学院 図書館情報メディア研究科　〒305–8550 茨城県つくば市春日 1–2
†† 筑波大学 図書館情報メディア系/知的コミュニティ基盤研究センター　〒305–8550 茨城県つくば市春日 1–2
††† JST さきがけ
E-mail: yui@slis.tsukuba.ac.jp, mori@slis.tsukuba.ac.jp, hiroki.sodeyama.2009b@mlab.info

**あらまし**　近年，テキストデータとそれと並存する構造データから構成されたWebデータが広く普及しつつある．このようなWebデータの具体例として，WikipediaとDBpediaデータの組が挙げられる．我々は，このようなWebデータに対する問合せ言語としてGradation問合せ言語を提案してきた．Gradationとは，キーワード問合せをベースとし，簡単な追加記述によって構造データに対する問合せ条件を表現することで，キーワード問合せと構造化問合せのシームレスな融合を実現する言語である．本稿では，Gradationの特徴に着目して，Gradationでより精確(precise)な問合せを作成することを支援する手法を提案する．これにより，完全な構造化問合せやハイブリッド問合せの記述が容易になる．実データを用いた実験では，本手法を利用することでSQLを知らない被験者の多くが精確な問合せを記述することができた．

**キーワード**　問合せ構築支援，構造化問合せ，キーワード問合せ，Web検索，構造データ

## 1. はじめに

近年，テキストデータとそれと並存する構造データから構成されたWebデータが広く普及しつつある．このようなWebデータの例として，Wikipediaのページとそれと並存するDBpedia [1] のデータ の組が挙げられる．DBpediaはWikipediaの各記事に関する様々な情報をRDF形式の構造データとして提供している．もう1つの例としては，ACM Digital Library [2] のページと，DBLP RKB Explorer [3] のデータとの組が挙げられる．ACM Digital Libraryはコンピュータサイエンス分野の論文の書誌情報をテキストページで提供しており，DBLP RKB Explorerは，コンピュータサイエンス分野の論文書誌データベースであるDBLPのデータをRDF形式で持っている．各書誌情報のページとRDFデータ間の関係は論文に付与されたDOIで表される．SemanticWebビジョンにおけるLinked Open Dataの取り組み [4] 等と相まって，テキストデータとそれと並存する構造データから構成されたWebデータは今後ますます一般的に利用されると考えられる．

これまで，先述のようなWebデータに問合せを行う方法には，2つの選択肢しか存在しなかった．すなわち，**キーワード問合せ** (本稿では，一つ以上のキーワードと，それらをつなぐ論理演算子，及び括弧で記述された問合せをキーワード問合せと呼ぶ) を用いるか，もしくは**構造化問合せ** (SPARQLなどの構造化問合せ言語に従った問合せ) を用いるか，のどちらかである．キーワード問合せは，カジュアルユーザに普及しているが，表現力が低く，Webデータのセマンティクスを利用した問合せを記述することができない．一方，構造化問合せは，表現力が高く高度な問合せが可能であるが，言語の学習コストが大きく，また，問合せ対象のデータに関する知識も必要となるため，カジュアルユーザが習得・利用することは難しい．したがって，ユーザにとっては，テキストデータを対象としたキーワード問合せを行うか，構造データを対象とした構造化問合せを行うか，という二者択一の状況であった．

我々は，この問題に対する新しいアプローチとして，**テキストページ** (本稿では，テキストを含むWebページなどをテキストページと呼ぶ) と**グラフデータ** (本稿では，主にRDFデータを指す) の組からなるWebデータに対する問合せ言語である**Gradation問合せ言語** (Gradation Query Language. 以下，Gradation) を提案してきた [5] [6]．Gradationの設計目標は，キーワード問合せと構造化問合せの溝を埋めることによって，高度な問合せ能力をカジュアルユーザにとって身近なものにすることである．Gradationは，キーワード問合せと構造化問合せとのシームレスな融合を実現するために，キーワード問合せをベースとし，簡単な追加記述によって構造データに対する問合せ条件を表現する．Gradationにおいては，キーワードのみを使った問合せは通常のキーワード問合せのように動作し，構造データに対する問合せ条件のみを使った問合せは完全な構造化問合せとなり，キーワードと構造データに対する問合せ条件を混在させた問合せはテキストとグラフデータに対するハイブリッド問合せとなる．このような問合せ記述方法を採用することにより，Gradationは単純なキーワード問合せから高度な構造化問合せまでを広くカバーするとともに，“pay-as-you-go”スタイルの問合せを実現する．すなわち，問合せ記述にかけるコストや問合せ条件の精確さを，ユーザが自身の要求やスキルに応じて選択し，利用することを可能にする．しかし，Gradationで構造データに対する問合せ条件を記述する（つまりハイブ

リッド問合せや完全な構造化問合せを記述する）ためには，検索対象データのクラス名や属性名などのスキーマレベルの情報が必要であるため，SQL や SPARQL といった他の構造化問合せ言語を使用する場合と同じだけの知識が求められる．

本稿では，Gradation 問合せ作成を支援する手法 MorphingAssist を提案する．この提案手法は，Gradation の「キーワード問合せから構造化問合せまでを同一の言語でシームレスに記述できる」という特徴に着目した，より精確 (precise) な問合せの作成支援手法である．本支援手法の目標は，ユーザがキーワード問合せによる問合せから始め，その結果を見ながらキーワードを変更したり MorphingAssist が提示する**ヒント**を見ながら徐々に問合せを「必要な範囲で」「出来る範囲で」詳細化することを支援することである．本支援手法により，ユーザは完全な構造化問合せやハイブリッド問合せを容易に記述できる．

MorphingAssist の開発にあたっては，自明でない 3 つの問題に取り組む必要がある．**(1) 表示するヒント:** どのようなヒントを提示すればよいのか．**(2) ヒントの計算方法:** どのようにしてそのヒントを求めるか．**(3) ヒントの提示方法:** どのように求めたヒントをユーザに見せるべきか．詳細は 4. 節で説明するが，MorphingAssist の開発にあたり，我々は，(1) に関しては，一般的なキーワード検索の Suggest 機能と同様に,「もしかしたら，この (構造データに関する) 問合せ構成要素を使いたかったのかもしれない」という問合せ改訂案を見せるというアプローチを採用する．(2) に関しては，ユーザが文書検索を意図して入力したキーワードをあえて構造データとマッチさせ，ユーザが使用する可能性のある問合せ構成要素を計算するというアプローチを採用する．(3) に関しては，ヒントの組み合わせ爆発を避けつつ，ユーザが問合せの書き直しをしやすいと考えられる手法を採用する．実験の結果，提案アプローチは容易に実現可能であるにも関わらず効果的であることがわかった．

本稿の貢献は次の通りである．

**(1) 二択でない構造化問合せの作成支援.** 「完全な構造化問合せを出力できるか，そもそも問合せが出来ないかの二択」であった既存の SQL 言語等の構造化問合せ作成支援ツールと異なり，Gradation 問合せ言語を利用する事によって，キーワード問合せから徐々に「必要な範囲で」「出来る範囲で」問合せの詳細化を支援するという，新規性の高いアプローチを提案する．

**(2) 実データによる評価実験.** Wikipedia/DBpedia を用いて被験者を用いた評価実験を行った結果を示す．実験では，Gradation 言語と MorphingAssist を利用する事により，SQL 言語等の構造化問合せを書けない，もしくは書いたことがない被験者の多くが精確な問合せを記述することができた．

本稿の構成は次の通りである．2 節では，関連研究について述べる．3 節では，キーワード問合せと構造化問合せをシームレスに融合した問合せ言語 Gradation の概要を説明する．4 節で，Gradation を用いた構造問合せ作成支援 MorphingAssist を提案する．5 節では評価実験について述べる．6 節にまとめと今後の課題を示す．

## 2. 関 連 研 究

本研究は，キーワード問合せと構造化問合せをシームレスに融合した問合せ言語を利用して，ユーザによる構造化問合せの記述を支援する研究である．構造化問合せの記述を支援するシステムは数多く存在し，大きく次の 2 つに分類される．(1) ユーザに構造化問合せを入力させ，よりユーザの意図に合致すると思われる構造化問合せを構築していく支援方法．(2) ユーザにキーワードや自然言語文を入力させ，それを基に構造化問合せを構築する支援方法．

(1) の例としては，ユーザの書いた SQL 問合せを基に，別の SQL 問合せを推薦する研究 [8] が存在する．[8] では，ユーザは初めから構造化問合せを記述する必要がある．これに対して，我々の手法は，構造化問合せが記述できないユーザが構造化問合せを記述することを支援することを目的としており，ユーザは初めにキーワード問合せを入力するだけでよい．

(2) の例としては IQP [7] がある．IQP は最初にユーザに問合せを表すキーワードを入力させ，次にユーザに質問を行い，最終的にユーザの意図を反映した構造化問合せを出力する．本研究との違いは次の 2 点である．第 1 に，IQP は最終的な構造化問合せが出力するまで，ユーザが問合せを実行することが出来ない．これに対して，Gradation では，完全な構造化問合せでないキーワード問合せやハイブリッド問合せであっても実行することができる．これにより，ユーザが自身の要求やスキルに応じた程度の問合せを記述し実行可能である．第 2 に，IQP では，最初に入力されたキーワードが，最終的に出力される構造化問合せに大きく影響する．したがって，最初に適切なキーワードを入力できなければ，生成される問合せはユーザの意図を反映しなくなる可能性が高い．しかし，我々の経験から言うと，ユーザが最初から適切なキーワードを選ぶことは困難であることが多い．これに対して，Gradation を利用すれば，最初にキーワード問合せを実行したときから検索結果がその場で表示されるので，記述したキーワード問合せが不適切だと気付けば，その場で簡単にキーワードを変更することができる．

## 3. Gradation 問合せ言語

Gradation は，テキストページと構造データから構成された Web データを検索対象としたハイブリッド問合せ言語である．Gradation は，キーワード問合せをベースとし，簡単な追加記述によって構造データに対する問合せ条件を表現する言語であり，キーワード問合せと構造化問合せとのシームレスな融合を実現している言語である．

Gradation は，キーワード問合せのみを利用するようなユーザであっても，キーワードに加えて簡単な検索オプションをしばしば利用することに着目して設計されている．例えば，Web 検索エンジンのユーザは，指定したサイトのみを検索対象とするオプション (`site:ac.jp` など) を記述して検索することがある．実際, [9] によると，25%のユーザが検索オプションを使用したことがあると言われている．

Gradation でも，キーワードに加えて検索オプションのよう

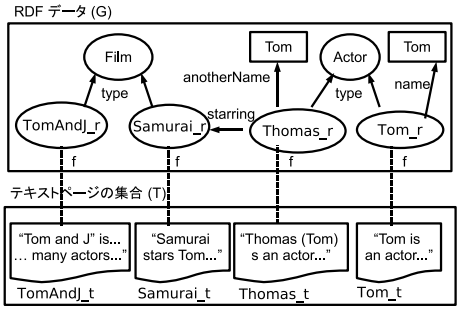


図 1　問合せ対象データ n の例

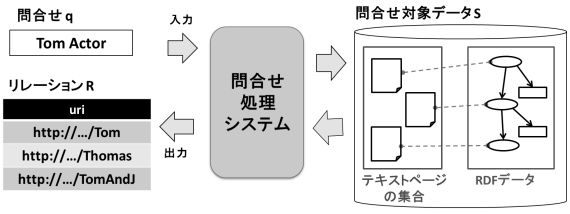


図 2　問合せ処理モデル

な簡単な追加記述を利用することができる．この簡単な追加記述によって構造化問合せの問合せ条件を表現する．例えば，グラフデータに各人物の年齢 (age) を表すデータが含まれる場合，40 歳以上の Tom という人物のテキストページを取得する問合せは，追加記述 `age>=40` を用いて `Tom age>=40` と記述する．

Gradation は単なるキーワード問合せや構造化問合せが記述できるだけでなく，ハイブリッドな問合せを記述することができる．ハイブリッドな問合せというのは，テキストデータの問合せ条件と構造データの問合せ条件を混合して記述している問合せである．

### 3.1　問合せ対象データ

Gradation の問合せ対象データ $D$ は，テキストページ $T$ とそれと並存するグラフデータ (RDF データ) $G$ の組である．図 1 は問合せ対象データの一例である．図 1 中の点線は，テキストページとノードの対応関係を表す関数 $f : T \rightarrow G$ を表す．

### 3.2　問合せ処理モデル

Gradation の問合せ処理モデルを図 2 に示す．入力は Gradation で記述した問合せ $q$ (図 2 左上)，問合せ対象データは 3.1 節で説明した問合せ対象データ $D$ (図 2 右)，出力は，特別な指定がなければ，$q$ にマッチするテキストページの URI の集合を表すリレーション $R$ (図 2 左下) である．

### 3.3　Gradation の問合せ構成要素と問合せ例

Gradation の問合せ構成要素 (C1 から C5) とそれを使った問合せ例を説明する．下記問合せ例では，図 1 を問合せ対象データとする．

**(C1) keyword:** あるキーワードを含むテキストページを探すための構成要素．例えば，Tom と Actor という文字列を含むテキストページを取得する問合せは次のようになる．

```
Tom Actor
```

**(C2) selection:** 次の条件を満たす RDF データと対応するテキストページを探すための構成要素．RDF データ中のノード ($value$) とエッジ ($attr$) の間に，“$attr\ \theta\ value$” もしくは “$attr_1\ \theta\ attr_2$” が成り立つ．$\theta$ には，等号・不等号の記号が使える．例えば，名前 (name) が Tom である俳優のテキストページを取得する問合せは次のようになる．

```
name==Tom type==Actor
```

**(C3) path traversal:** RDF データ中のノード間の関係 ($p$) を表すための構成要素．$q_1.p.q_2$ のように記述する．$q_1$ と $q_2$ には任意の構成要素を記述することができ，$q_1$ と $q_2$ を満たしかつ $p$ の関係にあるテキストページの組を探すための構成要素である．例えば，名前が Tom である俳優と，彼が主演を務めた (starringOf) 映画のテキストページの組を取得する問合せは次の通りである．

```
(name==Tom type==Actor).starringOf.type==Film
```

**(C4) projection:** 検索結果のリレーションのスキーマを指定するための構成要素．$q\ [attr_1, \ldots, attr_n]$ のように記述する．$q$ には任意の構成要素を記述することができる．$attr_i$ には RDF データ中のエッジの他，テキストページの URI を表す `t_uri` と RDF データのリソースノードを表す `r_uri` が記述できる．例えば，俳優を取得し，その俳優のテキストページ (t_uri) と名前 (name) を表示する問合せは次の通りである．

```
(type==Actor)[t_uri, name]
```

**(C5) and, or, not, *, ():** 任意の構成要素に対して，and, or, not, *(直積), グルーピングを行うことができる．例えば，俳優 (Actor) である人物のテキストページで，かつ Tom という文字列を含むテキストページを取得する問合せは次の通りである (and は省略可能である)．この問合せは，キーワード問合せの条件と構造化問合せの条件が混在した問合せである．

```
Tom  and  type==Actor
```

以上の問合せ例のように，Gradation を用いると，キーワード問合せの条件と構造化問合せの条件を混在させ，簡易なレベルの問合せから高度な問合せまでが表現可能である．

## 4.　MorphingAssist

3 節で述べたように，Gradation を利用すれば，キーワード問合せから構造化問合せまでを同一の言語でシームレスに記述できる．しかし，構造データに関する問合せ条件 (すなわち構成要素 C2 から C4) を記述するためには，対象データのクラス名や属性名などのスキーマレベルの情報が必要であるため，SQL や SPARQL といった他の構造化問合せ言語を使用する場合と同じだけの知識が求められる．

本節では，Gradation での精確な問合せ作成を支援する手法 **MorphingAssist** を提案する．これは，Gradation の「キーワード問合せから構造化問合せまでを同一の言語でシームレスに記述できる」という特徴に着目した，精確な問合せの作成支援手法である．MorphingAssist のユーザはキーワード問合せから始め，その結果を見ながらキーワードを変更したり，MorphingAssist の支援によって徐々に問合せを「必要な範囲で」「出来る範囲で」詳細化する．例えば,「俳優のトムという人物を全員知りたい」と思ったユーザが最初に問合せ (QI) “`Tom`

Actor” を記述したとする．もし MorphingAssist が適切な支援を行えば，ユーザは徐々に問合せをより精確な問合せに書き直して，(QII) “Tom type==Actor” や (QIII) “name==Tom type==Actor” を記述できる．MorphingAssist はこのような問合せの変更を支援するための**ヒント**を提示する．

MorphingAssist の開発にあたっては，自明でない 3 つの問題に取り組む必要がある．**(1) 表示するヒント:** どのようなヒントを提示すればよいのか．**(2) ヒントの計算方法:** どのようにしてそのヒントを求めるか．**(3) ヒントの提示方法:** どのように求めたヒントをユーザに見せるべきか．

**(1) 表示するヒント.** キーワード検索システムの問合せ推薦機能の一般的なアプローチは,「もしかしたら，このキーワードを使いたかったのかもしれない」と考えられる新しいキーワードを次の問合せのために推薦する，というアプローチである．我々はこのアプローチを我々の文脈に適用し,「もしかしたら，この (構造データに関する) 問合せ要素を使いたかったのかもしれない」と考えられる問合せ要素を次の問合せのために推薦するというアプローチを採用する．構造データに関する問合せ要素を推薦するので，元の問合せをより精確な問合せに書き直すことを支援できる．問合せの構成要素の例としては，“name==Tom” や “type==Actor” がある．この構成要素を使うと，QI を書き直して QII や QIII を書くことができる．

**(2) ヒントの計算方法.** ヒントを求めるために，我々はユーザが書いた元のキーワード (テキストデータとマッチすることを意図して書かれてたキーワード) をあえてそれと並存する構造データにマッチさせ，ユーザが使う可能性のある問合せ構成要素を計算する．この手法は単純であり，同時に，効果的であることが実験 (5 節参照) によって示されている．

**(3) ヒントの提示方法.** ヒントの単純な提示方法の 1 つは，元の問合せを書き換えるのに使うことのできる問合せの構成要素を並べて提示することである．しかし，これではユーザは提示された構成要素をどのように使えばいいか思いつくことができない．別の単純な方法としては，元の問合せと求めた構成要素を使って作成可能なより詳細な問合せへの改訂案 (以下，**問合せ改訂案**) を**全て**提示することである．しかし，これでは提示される問合せ改訂案の数は膨大になり，容易に組合せ爆発を起こしてしまう．

そこで，我々はヒントの組み合わせ爆発を避けつつ，ユーザが問合せの書き直しをしやすいと考えられる提示方法を提案する．その方法とは，元の問合せと 1 カ所だけ異なる問合せ改訂案を全て列挙することである．こうすることにより，ヒントの数は求めた構成要素の数と同じになり，かつユーザは提示されたヒントを組み合わせることで 2 つ以上の構成要素を合わせて使った新たな問合せを簡単に思いつくことができる．例えば，元の問合せが “Actor Film Star” であり，求めた構成要素が “type==Actor”，“type==Film”，“[star]”，“.star.” の 4 つだったとする．この時，MorphingAssist は次の 4 つのヒントを提示する．“type==Actor Film Star”，“Actor type==Film Star”，“(Actor Film)[star]”，“(Actor Film).star.(*)”.

MorphingAssist では，直前に投げた問合せの検索結果の 1 タプル毎にヒントが作成される．例えば，ユーザが問合せ QI を図 1 のデータに対して投げると，図 3 のように検索結果とヒントが表示される．我々のユーザインタフェースでは，ユーザは各タプルの隣にあるボタンをクリックするとそのタプルに関連するヒントが表示されるようにしている．これは，ユーザの関心のない検索結果 (タプル) に関連するヒントはユーザに必要ないと考えたためである．図 3 の検索結果と hint を見たユーザは，1 行目のヒントを見れば問合せ QII が書けるし，1 行目と 2 行目のヒントを組み合わせて使えば問合せ QIII が書ける．なぜなら，1 行目と 2 行目のヒントは元の問合せの “Tom” は “name==Tom” に置き換えることができて，“Actor” は “type==Actor” に置き換えることができることを示しているからである．

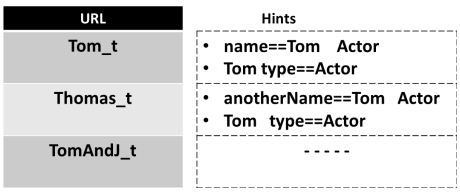


図 3　検索結果とヒントの例

さらに，適切そうなヒントを選び出すために，我々はこの提案手法を次の単純な手法と組み合わせる．まず，ユーザに，情報要求にマッチしているタプルを選ばせる．次に，それらのタプルと関連するヒントの積集合を計算し，ユーザに見せる．こうすることで，ユーザの情報要求を問合せで表現する際に重要なヒントがこの中に含まれることになると考えられる．

### 4.1　ヒントの生成

ヒントの生成は次の 2 段階からなる.【Phase1】使われる可能性のある全ての問合せ構成要素を求める.【Phase2】問合せ構成要素を使ってヒントを生成する．

**【Phase1】使われる可能性のある全ての問合せ構成要素を求める**．Gradation 問合せ処理システムによって求めた $q_i$ の検索結果を $R$ とする.【Phase1】では，検索結果 1 件 $r$ ($r \in R$) 毎に，問合せ改訂案 $q_{i+1}$ で使われる可能性のある構成要素を求める．すなわち，図 3 で言うと，各タプル毎に構成要素は求められる．具体的には,【Phase1】は次の 2 ステップからなる．

**Step 1:** 検索結果 1 件 $r$ ($r \in R$) に含まれるテキストページ $t$ の URI(検索結果のうち属性 t_uri) の集合を $U$ とする．このとき， $V = \{f(t)|t \in U\}$ となるような URI の集合 $V$ を求める．ここで，関数 $f(t)$ は $t$ と対応する RDF ノードを返す関数である (3 節参照).

**Step 2:** $V$ 中の RDF ノードから距離 $h$ 以内にある **RDF ノード**の集合を $V'$ とする．また，$V'$ を求める際にトラバースしたエッジのエッジラベルの集合を $L$ とする．このとき，次のルールに基づいて構成要素を生成する．

Rule1:　問合せ $q_i$ 中のキーワード 1 つが $V'$ 中のノードのノードラベルとマッチしたら，selection 構成要素 (C2) “$attr = value$” を生成する．ここで， $value$ はマッチしたノードラベルの値であり，$attr$ はそのノードから出ているエッジのエッジラベルである．もしマッチしたノードに対してそのノードから出ている

エッジが複数存在したら，それぞれのエッジに対して 1 つ構成要素を生成する．加えて，もし *value* が数字であった場合は，不等号記号を使った selection 構成要素 (“*attr* <= *value*” など) も生成する．

Rule2: 問合せ $q_i$ 中のキーワード 1 つが $L$ 中のエッジラベルとマッチしたら， projection の構成要素 (C4) “[$l$]” と path-traversal の構成要素 (C3) “.$l$.” を生成する．ここで，$l$ はマッチしたラベルである．

例えば，図 3 の 1 タプル目の検索結果 (Tom_t) に対応する構成要素は次のようにして求める．$q_i$ は “`Tom Actor`” で，ヒントを求めたい検索結果 (1 タプル目) は `t_uri` 属性に Tom_t を持っている．このとき，図 1 では $f$(Tom_t) = Tom_r が成り立つ．図 1 では，Tom_r の隣接ノードでキーワード “Tom” とマッチするノード (“name” エッジで繋がっているノード) とキーワード “Actor” とマッチするノード (“type” エッジで繋がっているノード) がある (ここで，“Tom” と “Actor” は問合せ $q_i$ に含まれているキーワードである)．従って，図 3 中の 1 タプル目の検索結果 (Tom_t) に対応する構成要素は “`name==Tom`” と “`type==Actor`” の 2 つとなる．

**【Phase2】問合せ構成要素を使ってヒントを生成する**．ここでは，(1) で生成した問合せ構成要素の 1 つと $q_i$ 中のキーワード 1 つを置き換えることでヒントを生成する．例えば，$q_i$ が `Tom Actor` で 1 件目の検索結果に対して生成された問合せ構成要素が “`name==Tom`” と “`type==Actor`” の 2 つだった場合，(2) で生成するヒントは “`name==Tom Actor`” と “`Tom type==Actor`” の 2 つである．

## 5. 評 価 実 験

本節では MorphingAssist の評価実験について述べる．本実験では次の 2 つの観点から評価を行った．(1) 生成したヒントの質．(2) MorphingAssist の効果．本実験では，Gradation と MorphingAssist の組合せは効果的であった．また，本実験では SQL を知らない被験者の多くが精確な問合せを記述できた．

### 5.1 設　　定

本実験で使用したデータは，Wikipedia [10] と DBpedia Japanese [11] のダンプデータである．

本実験では，2 つのグループの被験者に別々の作業をしてもらい実験を行う．**グループ A** : 実験で使用する情報要求を作成する学生 (7 名)．**グループ B** : MorphingAssist システムを使って Gradation 問合せを記述する学生 (10 名)．さらにグループ B は，グループ B1 とグループ B2 に分けることができる．グループ B1 は DB 言語 (例えば SQL や XQuery，SPARQL など) をある程度使うことができる学生 (5 人) であり，グループ B2 は DB 言語を 1 つも知らない，もしくはかつて習ったことがあるが忘れてしまった学生 (5 人) である．グループ B 全員は Gradation の文法に関する知識がなかったため，実験前に文法の説明を行った．

### 5.2 方　　法

本実験は次の手順で行った．

**(1) 情報要求と正解問合せの作成:** まず，グループ A の 7 人の被験者に情報要求 (Wikipedia や DBpedia に載っている情報に対する情報要求) を考えてもらった．この情報要求の集合を $N$ とする．次に，我々がそれぞれの情報要求 $n_i \in N$ を満たすような正解 Gradation 問合せ $cq_i$ を作成した．正解 Gradation 問合せの集合を $CQ$ とする．

| 要件 | 情報要求の例 |
|---|---|
| 物事の「属性」を使う | 「芥川賞を受賞した」作家を全員知りたい． |
| 複数の物事間の「関係」を使う | お笑い芸人と東京都に本社がある事務所の組み合わせ（「その芸人はその事務所に所属している」）全てが知りたい． |
| 物事の「属性」を表示する | 日本にある全ての動物園と「その開園年」が知りたい． |

図 4　情報要求作成に係る 3 つの要件と情報要求の例

**(2) 正解問合せの構成要素を求める:** それぞれの正解 Gradation 問合せ $cq_i \in CQ$ ごとに，その問合せの構成要素の集合 $C_i$ を求めた．例えば，$cq_i$ = “`name==tom type==actor`” のとき，$C_i$ = {`name==tom`, `type==actor`} である．

**(3) 情報要求から Gradation 問合せを記述:** グループ B の被験者に情報要求を提示し，MorphingAssist システムを使用して Gradation 問合せを記述してもらう．ここで，問合せ $q_{i,j,k}$ は，情報要求 $n_i \in N$ を見た被験者 $s_j$ が書いた $k$ 番目の問合せを表すこととする．$Q_{i,j}$ は， $q_{i,j,k}$ を $k$ を 1 から並べたリストとする．以降，$Q_{i,j}$ を，**問合せ記述プロセス** と呼ぶ．$q_{i,j,k}$ は，$Q_{i,j}$ の $k$ 番目の問合せとなる．また，問合せ $q_{i,j,k}$ の結果と共に被験者 $j$ に提示したヒントの集合を $H_{i,j,k}$ で表現する．

**(4) ヒントの質と MorphingAssist の効果から MorphingAssist を評価:** 4. 節に示したように， MorphingAssist は，ユーザに再検索で使われる可能性のある問合せ構成要素を含む問合せをヒントとして提示する．本実験では，次の 2 つの観点で MorphingAssist を評価する．(1) 生成したヒントの質 : ユーザに提示したヒントの表示数 $|H_{i,j,k}|$，ヒントの再現率 (正解問合せの構成要素 $C_i$ をどれだけ再現できたか)．(2) MorphingAssist の効果 : 被験者が記述した問合せの実行結果の再現率と精度，被験者が記述した問合せと正解問合せの比較．

#### 5.2.1 情報要求と正解問合せの作成

まず，グループ A の被験者それぞれに情報要求 (Wikipedia や DBpedia に載っている情報に対する情報要求) を 3 つ自然言語で記述してもらった．このとき，各情報要求が図 4 に書いてある要件 3 つのうち少なくとも 1 つを満たし,「○○なものを全部知りたい」という形の情報要求になるように指示をした．これは，Gradation における構造化問合せ構成要素のうち，基本的な構成要素が次の 3 つであるためである．(1) 属性を指定する構成要素 (C2)，(2) 関係を指定する構成要素 (C3)，(3) 検索結果に表示する属性名を指定する構成要素 (C4)．情報要求を作成してもらうにあたって，被験者には情報要求のサンプルとして図 4 の情報要求の例を示した．

次に，集めた情報要求の中から，DBpedia 中に情報がないために調べることができない情報要求を排除した．その結果，10 個の情報要求が得られた (つまり，$|N| = 10$)．

最後に，我々がそれぞれの情報要求 $n_i \in N$ を満たすような Gradation 正解問合せ $cq_i \in CQ$ を記述した．各 $cq_i$ の実行結

果を，情報要求 $n_i$ を記述した被験者に見せ，その結果が $n_i$ を満たしているかどうか確認してもらった．満たさない，と判断された場合は，満たすと判断されるまで $cq_i$ を書き直した（本実験では，満たさない，と判断されることはなかった）．付録 1. に，本実験で使用した情報要求とそれを満たす正解 Gradation 問合せを示す．

**5.2.2** 正解問合せの構成要素の作成

まず，各正解問合せ $cq_i \in CQ$ から問合せの構成要素を抽出して，$C_i$ ($C_i = components(cq_i)$) を作成する．次に，$C_i$ から，単純な問合せ構成要素を削除する．単純な問合せ構成要素とは，RDF データ中のデータ (エッジラベル，ノードラベル) を含まないような構成要素である．具体的には，(C5) の “`and`”，“`or`”，“`not`”，“`*`” である．これらを削除した理由は，MorphingAssist は，RDF データ中のデータに関する情報をヒントとして提示するシステムであり，これらの構成要素が MorphingAssist によって提示されないからである．例えば，$cq_i$ が “`(name==Tom or name==Thomas)`” であったとき，$C_i$ は {`name==Tom`, `name==Thomas`} となる．

**5.2.3** 被験者による問合せ記述

それぞれの情報要求をグループ B の被験者 3 人に見せ，MorphingAssist システムを使って Gradation 問合せを記述してもらった．各被験者 $s_j$ は 3 つの情報要求に対して Gradation 問合せを記述した．各被験者は，自分が書いた問合せ $q_{i,j,k}$ が情報要求 $n_i$ を満たすと判断するまで問合せを書き直すことができる．被験者が自分で情報要求を満たすと判断した問合せを**確定問合せ**と呼び，$dq_{i,j}$ と表す．被験者が 15 分以内に確定問合せを作成することができなかった場合は，確定問合せはなしとする．全ての被験者について最後の問合せ $q_{i,j,|Q_{i,j}|}$ は存在し，確定問合せが存在する場合は，$dq_{i,j} = q_{i,j,|Q_{i,j}|}$ となる．

### 5.3 実験結果 1：表示したヒントの質

図 5 中の 1 つの点は問合せ記述プロセス 1 つ $Q_{i,j}$ を表す．ただし，ユーザが全くヒントを見なかった 2 プロセスは除く．y 軸は $|\cup_k H_{i,j,k}|/|Q_{i,j}|$，つまり各問合せ記述プロセス内で表示したヒントの数 (被験者がヒントを表示しなかった場合は除く) の平均を表す．$\cup_k$ は bug-union を求める記号を表し，$\cup_k H_{i,j,k}$ は多重集合である．x 軸は $C_i$ に対する ヒントの再現率を表す．ヒントの再現率とは，問合せ記述プロセス中で正解構成要素がどれだけ表示されたかを表すもので，$|C_i \cap \cup_k H_{i,j,k}|/|C_i|$ によって求められる．ヒントの再現率の平均は 0.71 であり，ヒントの表示数の平均は 25.76 であった．本評価観点で精度を求めないのは，確定問合せでない問合せが書かれた意図は一般的に不明であり，特定の意図をもって書かれた正解問合せに対する精度を求めることは意味をなさないからである．図 5 の全 28 プロセス中，4 プロセスで，再現率は 0.4 以下であった．しかし，これは被験者が MorphingAssist システムを使わなかったことが原因であった．

本実験では，30 問合せプロセス中，10 プロセスで，積集合を用いてヒントの表示数を減らす仕組み (4 節参照) が使われた．この 10 プロセスにおいて，ヒントの表示数を減らす仕組みを使う前と後で，ヒントの表示数の平均は 43.22 から 9.71 に減

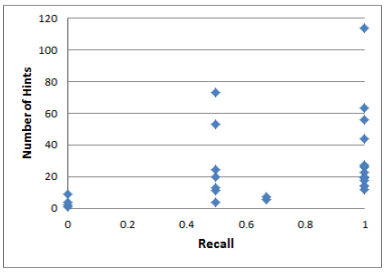


図 5 提案手法による構成要素の再現率とヒントの表示数

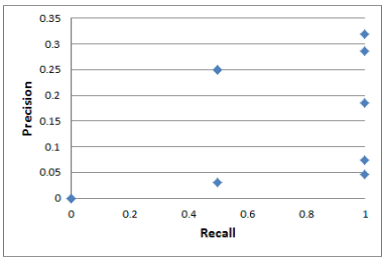


図 6 ヒントを減らす工夫をした時のヒントの再現率とヒントの精度

少したにもかかわらず，再現率が下がったものは 1 つもなかった (再現率の平均は 0.65)．

被験者がヒントを減らす仕組みを使うということは，被験者の意図を表しているので，ここで精度を求めることは意味がある．図 6 は，ヒントを減らす仕組みを使用した後に表示されたヒントの再現率と精度である．ヒントの精度は，$|C_i \cap \cup_k H_{i,j,k}|/|\cup_k H_{i,j,k}|$ によって求められる．ヒントの精度の平均は 0.16 であった．これは，表示されたヒント平均 9.71 個のうち 1.55 個は正解構成要素を含んでいることを示す．

以上の結果から， MorphingAssist は適切なヒントを表示することに関してよいはたらきをしていることが分かる．

### 5.4 実験結果 2： MorphingAssist の効果

本節では， MorphingAssist の効果について次の 2 つの観点から評価する．(1) 確定問合せの検索結果の再現率と精度．(2)MorphingAssist の支援によりユーザが書き直した問合せがより精確なものになっているか．

**5.4.1** 確定問合せの検索結果の再現率と精度

図 7 に，正解問合せの検索結果に対する確定問合せの検索結果の再現率と精度を示す．再現率は $|Result(dq_{i,j}) \cap Result(cq_i)|/|Result(cq_i)|$ であり，精度は $|Result(dq_{i,j}) \cap Result(cq_i)|/|Result(dq_{i,j})|$ である．図 7 より多くの被験者は正しい問合せを記述することができたことが分かる．着目すべき点は，被験者のうちの半分は SQL を書くことができない人であったことである．

**5.4.2** 記述した問合せの変化

次に，本節では確定問合せに向けて問合せがどのように変化したかを評価する．これは，前節で示した結果が本当に MorphingAssist によるものかどうかを確認するためである．

**問合せのスコア**．同じ問合せ記述プロセス内の問合せであっても，問合せ $q_{i,j,k}$ 同士を比較することは難しい．なぜなら，各問合せの検索結果は異なるスキーマ構造を持っており，再現率と精度を計算することが困難であるからである．従って，我々は各 $q_{i,j,k}$ 毎に，その問合せがどれだけ正解問合せ $cq_i$ と違っている

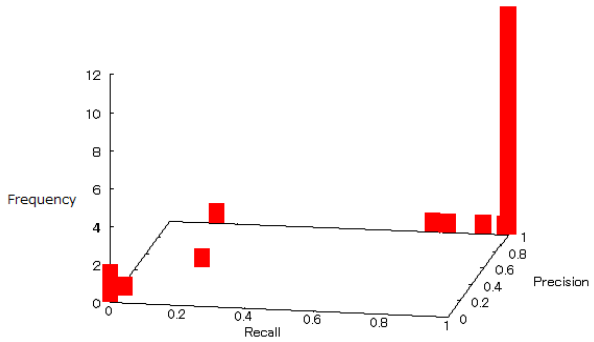


図 7　確定問合せの検索結果の再現率と精度

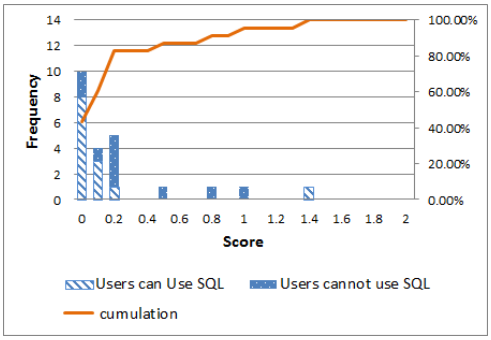


図 8　確定問合せのスコアの分布

かを表すスコア (difference score) を付けることを考えた．このスコアは関数 $ds_{cq}(q)$ で表す．このスコアは問合せ間の違いが小さいほど値が小さい．すなわち，$q = cq$ の時，$ds_{cq}(q) = 0$ であり，$q$ と $cq$ の違いが大きければ大きいほど $ds_{cq}(q)$ は大きくなる．スコアの求め方の詳細は付録 2. に示す．

問合せ記述プロセス毎に，被験者が最初に書いた問合せ，確定問合せ，最後の問合せのスコアを求めた．最初に書いた問合せのスコアが 1 になるように正規化すると，23 個の確定問合せのスコアの平均は 0.199 であった．最後の問合せにおいても，スコアの平均は 0.310 であった．このことから，各被験者は最終的には正解問合せに近い問合せを記述できたことが分かる．

図 8 と 9 にスコアの詳細を示す．この図は，確定問合せ (図 8) と最後の問合せ (図 9) のスコアのヒストグラムと累積度数分布である．確定問合せのうち 82%はスコアが 0.2 以下であった．また，SQL を使うことができない被験者によって書かれた確定問合せのうち 70%はスコアが 0.2 以下であった．更に確定問合せでないものも含む全ての最後の問合せに関しても，その 70%はスコアが 0.2 以下であった．まとめると，これらの実験結果は，平均的には，MorphingAssist がユーザがより精確な問合せ (正解問合せに近い問合せ) を記述することの支援に成功していることを示している．図 9 では，最後の問合せに対するスコアを示したが，実際には，ユーザが書き直した問合せが適切でないと感じた場合は，ユーザは検索の際には元の問合せを使えばよく，最後の問合せを使う必要はない．

## 6.　まとめと今後の課題

本稿では，キーワード問合せと構造化問合せをシームレスに融合した問合せ言語を利用して，ユーザによる構造化問合せの記述を支援する手法を提案した．具体的には，キーワード問合

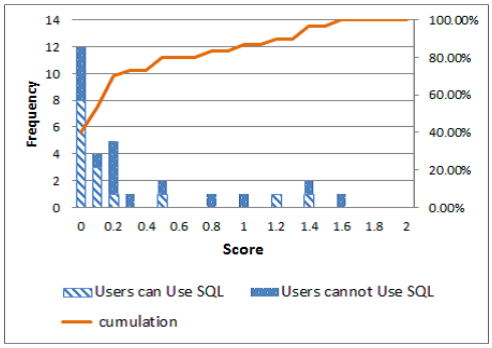


図 9　最後の問合せのスコアの分布

せを入力したユーザに対して，そのキーワード問合せを構造化問合せに変更するためのヒントを提示する．評価実験により，支援手法を使ったユーザはより精確な問合せが記述できることを示した．今後の課題としては，Gradation の検索結果およびヒントのランキング手法の開発が挙げられる．

## 謝　　辞

ゼミなどにおいてコメントいただきました，筑波大学 杉本重雄教授，阪口哲男准教授，永森光晴講師に感謝いたします．本研究の一部は JST さきがけ「情報環境と人」の支援による．

## 文　　献

[1] DBpedia: wiki.dbpedia.org : About, http://www.dbpedia.org/.
[2] ACM DIGITAL LIBRARY, http://dl.acm.org/.
[3] dblp.rkbexplorer.com, http://dblp.rkbexplorer.com/.
[4] W3C: LinkingOpenData, http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData.
[5] 袖山広輝, 只石正輝, 安永ゆい, 品川徳秀, 森嶋厚行. “構造/テキスト Web データのためのハイブリッド問合せ言語”. Web とデータベースに関するフォーラム 2011, 9 pages, (2011).
[6] Yui Yasunaga, Atsuyuki Morishima, Hiroki Sodeyama and Masateru Tadaishi. “Gradation: A Pay-as-You-Go Style Hybrid Query Language for Structured and Text Data”, Proceedings of the 2013 iConference (iConference 2013), pp.209-220, (2013).
[7] Demidova, E., Zhou, X. and Nejdl, W. “IQP: Incremental query construction, a probabilistic approach”, 26th IEEE International Conference on Data Engineering (ICDE 2010), pp.349-352, (2010).
[8] Chaitanya Mishra and Nick Koudas. “Interactive Query Refinement”, Proceedings of the 12th International Conference on Extending Database Technology (EDBT 2009), pp.862-873, (2009).
[9] internet.com K.K. (Japan) “An article from internet.com K.K.(Japan)”, http://japan.internet.com/wmnews/20090908/5.html(2009).
[10] Wikimedia Downloads, http://dumps.wikimedia.org/jawiki/20120603/.
[11] DBpedia Japanese, http://ja.dbpedia.org/dumps/20120629/.

## 付　　録

### 1.　実験で用いた情報要求と正解問合せ

実験で使用した情報要求とそれと対応する Gradation 問合せのリストは表 A·1 の通りである．

### 2.　スコア $ds_{cq}(q)$ の求め方

本付録では，ユーザが書いた問合せ $q$ と正解問合せ $cq$ が与えられた時に，$q$ と $cq$ の違いを表すスコア (difference score) を計算するための関数 $ds_{cq}(q)$ について説明する．この関数は 5.4.2 節の評価で利用したものであり，$q$ と $cq$ の違いが小さい

表 A·1　情報要求とそれと対応する Gradation 問合せのリスト

| ID | Information Needs<br>Correct Gradation Query |
|---|---|
| $n_1$ | 神奈川にある公園を全て知りたい<br>dcterms:subject == “category-ja:神奈川県の公園” |
| $n_2$ | 東京都内にある医療機関全てと，その医療機関の所在地が知りたい<br>(dcterms:subject == “category-ja:東京都の医療機関” )[t_uri, dbpprop-ja:所在地] |
| $n_3$ | ロンドンがホームタウンであるサッカークラブとそのクラブのホームスタジアムが知りたい<br>(dbpprop-ja:ホームタウン == “dbpedia-ja:ロンドン”) [t_uri, dbpprop-ja:スタジアム] |
| $n_4$ | 日本にある灯台を全て知りたい<br>dcterms:subject == “category-ja:日本の灯台” |
| $n_5$ | 日本にある女子高を全て知りたい<br>dcterms:subject == “category-ja:日本の女子高等学校” |
| $n_6$ | 日本のロックバンドと，そのバンドが所属しているレコード会社が知りたい<br>(dcterms:subject == “category-ja:日本のロック・バンド”) . dbpprop-ja:label . (dcterms:subject == “category-ja:日本のレコード会社”) |
| $n_7$ | 1990 年生まれのタレントを全員知りたい<br>dcterms:subject == “category-ja:日本のタレント” dbpprop-ja:生年 == 1990 |
| $n_8$ | 日本にある保守政党全てと，その政党の成立年が知りたい<br>(dcterms:subject == “category-ja:日本の保守政党”)[t_uri, dbpprop-ja:成立年月日] |
| $n_9$ | つくば市にある研究所を全て知りたい<br>dcterms:subject == “category-ja:つくば市の研究所” |
| $n_{10}$ | イタリアセリエ A に所属するサッカークラブ全てと，その監督名が知りたい<br>(dbpprop-ja:リーグ == “dbpedia-ja:セリエ A_(サッカー)”)[t_uri, dbpprop-ja:監督] |

**INPUT:** query q, correct query cq
**OUTPUT:** diffference score *score*
1: $score = 0$;
2: $rest = components(cq)$;
3: // penalty for each components を計算
4: **FOREACH** $c_i \in components(q)$
5: $cc_i = counterpart_{q,cq}(c_i)$
6: $score = score + c\text{-}penalty(c_i, cc_i)$;
7: $rest = rest$ - $cc_i$
8: **ENDFOR**
9: // penalty for missing components を計算
10: $score = score + m\text{-}penalty * |rest|$

図 A·1　$ds_{cq}(q)$ の計算

ほど値は小さくなる．すなわち，$q = cq$ の時，$ds_{cq}(q) = 0$ であり，$q$ と $cq$ の違いが大きければ大きいほど $ds_{cq}(q)$ は大きくなる．

スコア $ds_{cq}(q)$ は 図 A·1 で求められる．具体的には，まず $q$ 中の問合せ構成要素 $c_i \in components(q)$ に対して，それと対応する正解 $cq$ 中の構成要素 $cc_j \in components(cq)$ を見つけ，その 2 つがどれぐらい異なるかによってペナルティ(c-penalty)を加点する．次に，$cq$ 中の構成要素 $cc_j \in components(cq)$ のうち，先の作業で使われなかった要素の数だけペナルティ(m-penalty) を加点する．図 A·1 の 4-8 行目で，c-penalty を加点している．ここで，5 行目の $counterpart_{q,cq}(c_i)$ は，$c_i$ に対応する $cq$ 中の構成要素を求める関数である．この詳細は下記 2.1 節で述べる．また，6 行目の $c\text{-}penalty(c_i, cc_j)$ は構成要素の差異に応じた c-penalty を求める関数である．c-penalty の点数については下記 2.2 節で述べる．2,7,10 行目で m-penalty を加点している．10 行目の *m-penalty* は定数であり，今回は 5 としているが，この点数についても下記 2.2 節で述べる．

### 2.1　構成要素の対応付け関数

$counterpart_{q,cq}$ は，ユーザが記述した問合せ $q$ と 正解問合せ $cq$ のペアがある時に，$q$ 中の構成要素 $c_i$ が，$cq$ 中のどの構成要素の書き換えであるか（どの構成要素を意図して書いたか）を表す関数である．例えば，情報要求が「トムという俳優を全員知りたい」だった場合に，$q$ = “`tom actor`”, $cq$ = “`name==tom type==actor`”, $c_1$ = “`tom`”, $cc_1$ = “`name==tom`” がある時，$counterpart_{q,cq}(c_1) = cc_1$ が成り立つ．関数 $counterpart_{q,cq}$ は，$q$ の構成要素の集合 $components(q)$ を定義域とし，$cq$ の構成要素 $components(cq)$ と $nill$ からなる集合を値域とする．一般には，問合せ作成者の意図をくんで $counterpart_{q,cq}$ を計算するのは困難であるが，上記のように定義域と値域を限定する事により，「$cc_j$ が $c_i$ の類義語を含む」時に $counterpart_{q,cq}(c_i) = cc_j$ が成立し，そうでない場合は $counterpart_{q,cq}(c_i) = nil$ が成立する，というヒューリスティクスが利用可能な場合がある．今回の実験で利用した全ての問合せは，このヒューリスティクスが適用可能なケースであったため，これを利用して $counterpart_{q,cq}$ を計算した．

表 A·2　c-penalty の値

| type of $c_i$ | relation between $c_i$ and $cc_i$ | $\exists cc_j \in components(q)$ s.t. $c_j = cc_i$ | *otherwise* |
|---|---|---|---|
| keyword | $match(c_i, cc_i)$ | 1 | 3 |
| | $match(similar(c_i), cc_i)$ | 2 | 4 |
| | $otherwise\ (cc_i = nil)$ | 5 | 5 |
| non-keyword component | $result(c_i) = result(cc_i)$ | 0 | 0 |
| | $result(c_i) \supset result(cc_i)$ | 1 | 2 |
| | $result(c_i) \subset result(cc_i)$ | 3 | 3 |
| | $otherwise\ (cc_i = nil)$ | 5 | 5 |

### 2.2　ペナルティの値の決定

*c-penalty*$(c_i, cc_i)$ および *m-penalty* の値は次の原則に従い，それぞれ c-penalty を表 A·2，m-penalty=5 とした．

**原則：** **$c_i$ の存在によって $q$ の検索結果が $cq$ の検索結果により近くなる場合にペナルティの値はより小さくなるべきである．**

「検索結果に近い」ということを定めるために次の 3 つのヒューリスティクス (H1,H2,H3) を用いる．表 A·2 の行の場合分けに H1 と H2 を用い，列の場合分けに H3 を用いている．

**(H1)** $c_i$ がキーワードであるとき，(a)$cc_i$ がその一部に文字列 $c_i$ を含む場合，(b)$cc_i$ がその一部に文字列 $c_i$ の類義語を含む場合，(c)$cc_i$ が $nil$ である場合，の順に $q$ の検索結果は $cq$ の検索結果に近くなる．例えば，$q$ の構成要素 $c_i$ が `thomas` であるときよりも，`tom` であるときの方が，$q$ の検索結果は `name==tom` を含む正解問合せ $cq$ の検索結果に近くなると考えられる．

**(H2)** $c_i$ がキーワード以外の構成要素である場合は，(a) $result(c_i) = result(cc_i)$, (b) $result(c_i) \supset result(cc_i)$, (c) $result(c_i) \subset result(cc_i)$, (d)$cc_i = nil$ の順に $q$ の検索結果は $cq$ の検索結果に近くなる (ここで $result(c_i)$ は $c_i$ 単体の検索結果を表す)．例えば，`type==japaneseActor` よりも `type==allActor` が問合せに含まれる方が，`type==asiaActor` を含む問合せの検索結果に近くなると考えられる．

**(H3)** $\exists c_j \in components(q)$ s.t. $c_j = counterpart_{q,cq}(c_i)$ が成立するとし ($c_i$ と $c_j$ は異なってもよい)，$q'$ を $q - \{c_j\}$ とする．この時，$result(q)$ の方が $result(q')$ よりも $result(cq)$ に近くなる．例えば，$q$ = “`tom name==tom actor`”, $c_1$ = “`tom`”, $cq$ = “`name==tom type==actor`”, $cc_1$ = “`name==tom`” である場合を考える．この時，$q$ は $c_2 = cc_1$ であるような $c_2$ (つまり “`name==tom`”) を含む．更に，$q'$ = “`tom actor`”, $c_1'$ = “`tom`” であるとする．この時，$result(q)$ の方が $result(q')$ よりも $result(cq)$ に近くなる．