領域代数を用いた構造化テキスト検索の頑健でスケーラブルなモデル

辻井 潤一
大学院情報学環

概 要

構造化テキストに対する頑健でスケーラブルな近接構造検索モデルとその性能評価について報告する。本モデルでは、クエリに完全には一致していないが、部分的に一致しているテキスト範囲も検索することが可能であり、また、重みづけを与えることでテキスト範囲とクエリとの関連度を計算しランキング検索が行われる。クエリに対する関連度の近似計算を行うことにより、大規模文書集合に対する高速な検索が可能となり、スケーラビリティが改善される。実験では OHSUMED テストコレクションに対する精度を評価し、近似計算による高速化と精度低下のトレードオフも評価した。

1 はじめに

XML や領域代数を含めたテキスト検索の分野では、タグなどで示されるテキスト中の構造を指定した検索により従来のキーワード検索では検索できない情報を検索する手法が研究されてきている [1, 2, 4, 5] 。しかしながら、それらの手法は従来のキーワード検索のような頑健性をもたない、すなわちあるクエリを与えたときにそのクエリに対する完全一致のみを結果として出力する手法であったり、頑健であっても Web のような大規模文書集合に対しては適用できない手法であった。

本年度は、構造化テキストに対する頑健でスケーラブルな近接構造検索モデルの提案とその性能評価を行ったので、それらについて報告する。クエリから作られるサブクエリを用いることにより、クエリに完全には一致していないが、部分的に一致しているテキスト範囲も検索される。各サブクエリに対して重みを与えることでテキスト範囲とクエリとの関連度を計算しランキング検索を行う。またクエリに対する関連度の近似計算を行うことにより、大規模文書集合に対する高速な検索が可能となり、スケーラビリティが改善される。

$G_{q_1 \triangleright q_2}$	$=$	$\Gamma(\{a \mid a \in G_{q_1} \wedge \exists b \in G_{q_2}.(b \sqsubset a)\})$
$G_{q_1 \not\triangleright q_2}$	$=$	$\Gamma(\{a \mid a \in G_{q_1} \wedge \not\exists b \in G_{q_2}.(b \sqsubset a)\})$
$G_{q_1 \triangleleft q_2}$	$=$	$\Gamma(\{a \mid a \in G_{q_1} \wedge \exists b \in G_{q_2}.(a \sqsubset b)\})$
$G_{q_1 \not\triangleleft q_2}$	$=$	$\Gamma(\{a \mid a \in G_{q_1} \wedge \not\exists b \in G_{q_2}.(a \sqsubset b)\})$
$G_{q_1 \triangle q_2}$	$=$	$\Gamma(\{c \mid c \sqsubset (-\infty, \infty) \wedge \exists a \in G_{q_1}.\exists b \in G_{q_2}.(a \sqsubset c \wedge b \sqsubset c)\})$
$G_{q_1 \triangledown q_2}$	$=$	$\Gamma(\{c \mid c \sqsubset (-\infty, \infty) \wedge \exists a \in G_{q_1}.\exists b \in G_{q_2}.(a \sqsubset c \vee b \sqsubset c)\})$
$G_{q_1 \diamond q_2}$	$=$	$\Gamma(\{c \mid c = (p_s, p'_e) \text{ where } \exists (p_s, p_e) \in G_{q_1}.\exists (p'_s, p'_e) \in G_{q_2}.(p_e < p'_s)\})$

表 1: 領域代数の演算

2 背景:領域代数

領域代数 [2, 4] は開始位置と終了位置の組で表現される**領域**の集合と、領域の集合に対する演算により定義される。この領域代数を用いることによって、テキスト中の構造を指定した検索が可能となる。

本研究は [2] で提案されている領域代数を基にしている。この領域代数は以下の 7 個の演算子からなる。

- 包含演算子 ($\triangleright$, $\not\triangleright$, $\triangleleft$, $\not\triangleleft$):二領域間の包含関係
- 結合演算子 ($\triangle$,$\triangledown$):二領域の組合せ (AND, OR)
- 順序演算子 ($\diamond$):二領域の順序関係

二領域間の包含関係は次のように表現される。領域 $r = (p_b, p_e)$ が領域 $r' = (p'_b, p'_e)$ を含む

$\Leftrightarrow p_b \leq p'_b \leq p'_e \leq p_e$($p_b$:開始位置、$p_e$:終了位置)

この関係を $r \sqsubset r'$ で表す。Clarke らによる領域代数 [2] では、演算結果としての領域集合の中に包含関係を満たす領域が存在する場合には最も内側の領域のみ返すように定義されている。これは領

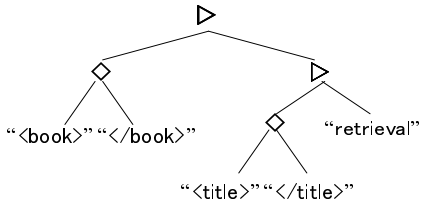

図 1: クエリ ‘[book] ▷ ([title] ▷ “retrieval”)’ の木構造表現

域の集合 S に対する関数 Γ

$$\Gamma(S) = \{r | r \in S \wedge \nexists r' \in S.(r' \neq r \wedge r' \sqsubset r)\}$$

で定義される。この関数 Γ を用いることで、上の演算子による演算の結果は表 1 のように表現される。

領域代数を用いたクエリは、図 1 の木構造で表現される。領域代数を用いたクエリに一致する領域を検索するアルゴリズムは、そのクエリ中に現れる単語で最も頻度が低い単語の頻度に線形の計算量であり、高速に検索を行うことができる。

3 頑健性の改善

構造化テキスト検索の頑健性を改善するために、ユーザが与えるクエリから作られるサブクエリを用い、そのサブクエリに一致する領域を含む文書も出力する。また文書のクエリに対する関連度を計算し、それを基にランキングを行う。

3.1 検索モデル

検索の頑健性を改善するため、本モデルでは与えられたクエリから作られるサブクエリを用い、クエリに部分的に一致している領域も出力する。サブクエリは以下のように定義する。

定義 1 (サブクエリ) クエリ q の木構造表現における部分木をサブクエリ $q_1, q_2, ..., q_n$ とする。

文書とクエリの関連度は、文書と各サブクエリとの関連度を基に計算する。文書とサブクエリの関連度は、サブクエリのその文書での重みと、サブクエリ自身の文書集合での重みを用いて計算される。

定義 2 (文書とサブクエリの関連度) サブクエリ q_i の文書 d での重みを $w_{q,d}$、q_i の文書集合での重みを w_{q_i} とする。q_i と d の関連度 $\phi(q_i, d)$ は関連度を表す関数 μ を用いて次のように定義される。

$$\phi(q_i, d) = \mu(w_{q_i,d}, w_{q_i})$$

文書と各サブクエリの関連度が計算された後、それを基に文書とクエリの関連度が計算される。

定義 3 (文書とクエリ全体の関連度) サブクエリ q_i と文書 d の関連度を $\phi(q_i, d)$ とする。d とクエリ q の関連度 $\Phi(q, d)$ はクエリ全体との関連度を表す関数 ρ を用いて以下のように定義される。

$$\Phi(q, d) = \rho(\phi(q_1, d), \phi(q_2, d), ..., \phi(q_m, d))$$

3.2 実装

本論文ではキーワードによる検索で用いられる TFIDF 値を構造付クエリに拡張した値を重みとして使用する。文書とクエリの関連度には cosine 値を用いる。

定義 4 (クエリの TF 値、IDF 値) 文書 d 中のクエリ q と一致する領域の数を $freq_q(d)$、クエリ q と一致する領域を含む文書数を df_q、全体での文書数を N とする。クエリ q の文書 d における TF 値および IDF 値を以下のように定義する。

$$tf_{q,d} = \begin{cases} 1 + \log(freq_q(d)) & (if\ freq_q(d) \neq 0) \\ 0 & (if\ freq_q(d) = 0) \end{cases}$$

$$idf_q = \begin{cases} \log(\frac{N}{df_q}) & (if\ df_q \neq 0) \\ 0 & (if\ df_q = 0) \end{cases}$$

これらの値を用いて文書とサブクエリ、およびクエリ全体との関連度を以下のように定義する。

$$\phi(q_i, d) = \mu(w_{q_i,d}, w_{q_i}) = tf_{q,d}\ idf_q$$

$$\Phi(q, d) = \frac{\sum_{i=1}^{n} \phi(q_i, d)}{\sqrt{\sum_{i=1}^{n} tf_{q,d}^2}\sqrt{\sum_{i=1}^{n} idf_q^2}}$$

3.3 実験

一般の情報検索の評価に使われる OHSUMED テストコレクション [3] を評価に用いる。OHSUMED は、348,566 個の医学論文のアブストラクトと、106 個のクエリからなる。各クエリに対し関連があるアブストラクトおよび関連があるか判定されたアブストラクトの集合が与えられている。

1	‘ “postmenopausal” △ ([neoplastic] ▷ (“breast” ◇ “cancer”)) △ ([therapeutic] ▷ (“replacement” ◇ “therapy”)) ’ 55 year old female, postmenopausal does estrogen replacement therapy cause breast cancer

表 2: クエリ例

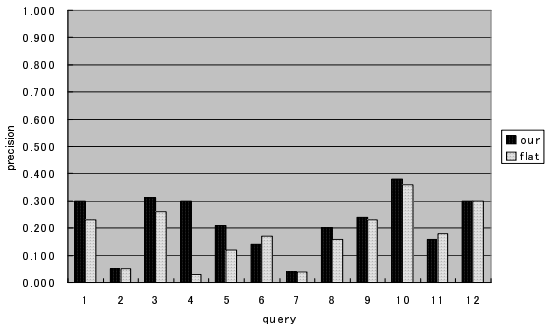

図 2: 適合率 (*our*、*flat*)

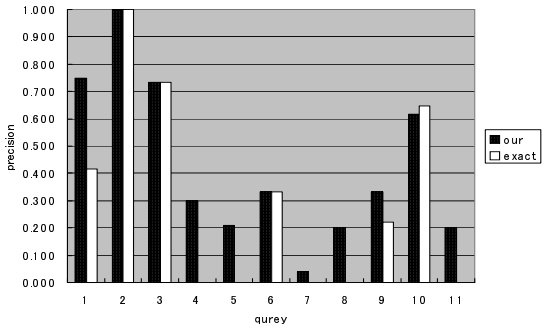

図 3: 適合率 (*our*、*exact*)

OHSUMED テストコレクションのクエリは自然言語で書かれているため、専門家が領域代数に変換した 12 個のクエリを用いた。使用したクエリの例を表 2 に示す。一行目が領域代数に変換したクエリ、二、三行目が元の OHSUMED のクエリである。また OHSUMED の検索対象のアブストラクトには文書構造は付与されているものの、意味タグは付与されていないため、専門用語の最長一致を用いて意味タグを付与した。

本モデル (*our*)、領域代数 (*exact*)、キーワード検索 (*flat*) の 3 つのモデルについて実験を行った。本モデルとキーワード検索については上位 100 文書、領域代数については完全一致した文書をすべて取り出し、適合率、再現率を比較した。その結果を図 2、3、4 に示す。領域代数 (*exact*) と本モデル (*our*) を比べると、再現率は本モデルのほうが高く領域代数では検索できない文書が検索できており、頑健性が改善されていることがわかる。また適合率に関しては、領域代数での検索結果がある場合にはそれほど変わらない。またキーワード検索 (*flat*) と本モデル (*our*) を比較すると、クエリによって差はあるものの全体としては適合率、再現率ともに本モデルが上回っている。

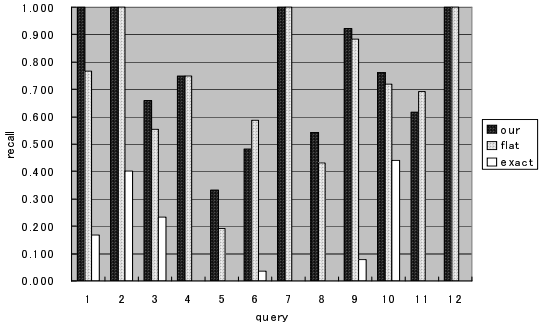

図 4: 再現率

4 フィルタリング

膨大な文書集合を検索対象とする場合、全文書に対してクエリとの関連度計算を行うことは非常に高コストである。関連度の近似値を用いて関連度が高くなると考えられる文書のみを選び出し、それらの文書に対してのみ関連度計算を行うことで検索時間を短縮する。従って、関連度の近似値は以下の性質備えていることが望ましい。

- 近似値が高ければ関連度も高い
- 近似値が高い文書への検索が高速に行える

近似値がこれらの特徴を持つ場合にはフィルタリングの効果が高くなり、特に関連度が高い文書に対してのみスコア計算を行い、検索時間が大幅に短縮される。

4.1 TFIDF 法のためのフィルタリングスコア

フィルタリングには、領域代数の完全一致を利用する。領域代数の完全一致の検索は高速であり、ある文書にサブクエリの完全一致が存在すれば関連度で用いている TF 値が高くなるため、関連度も高くなる。さらに、すべてのサブクエリを用いるのではなく重みの高いサブクエリのみを用いることでさらに検索時間を短縮する。

```
function Retrieve(q): ranking list;
    begin
        Q := SelectSubquery(q,v);
        S := EvalExact(Q);
        foreach d ∈ S
            begin
                r := CalcRel(d,q);
                PushRanking(L,d,r);
            end
        return L;
    end
SelectSubquery: 重みが v を超えるサブクエリの集合を返す
EvalExact: Q 中の各サブクエリと一致する領域を含む文書の
           集合を返す
CalcRel: 文書 d とクエリ q の関連度を返す
PushRanking: 文書 d と関連度 r の組をランキングリスト L
             に入れる
```

図 5: 検索アルゴリズム

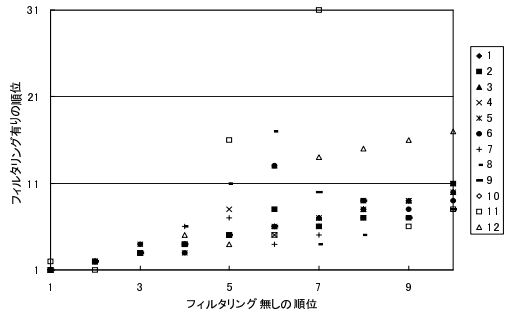

図 6: フィルタリングによる上位 10 文書の順位変化

ここでは関連度の近似値として以下のフィルタリングスコア $\phi'(q_i, d)$ を用いる。

$$\phi'(q_i, d) = \begin{cases} idf_q & (d \text{ が } q_i \text{に一致する領域を含む場合}) \\ 0 & (d \text{ が } q_i \text{に一致する領域を含まない場合}) \end{cases}$$

フィルタリングを用いた検索アルゴリズムを図 4.1 に示す。正確な IDF 値を計算するにはすべての文書を見なければならず高コストであるので、無作為抽出を行って近似的に計算した IDF 値を重みとして用いる。

4.2 実験

フィルタリングの有無による検索時間は 12 クエリの平均で次のようになった。

フィルタリング有り：0.20 秒

フィルタリング無し：8.63 秒

フィルタリングにより、検索時間が短縮されている。

また、フィルタリングのによる文書の順位の変化を図 6 に示す。無作為抽出を行ったことによる IDF 値の違いによって順位が多少変化しているものの、フィルタリングにおいて選び出されないということは起こっていない。

5 まとめと今後の課題

本年度は、頑健でスケーラブルな近接構造検索モデルの提案とその性能評価を行った。クエリから作られる細かなサブクエリを利用することで頑健性を改善し、近似値によるフィルタリングを行うことでスケーラビリティを改善した。

今後の課題としては精度の向上が考えられる。現在はキーワード検索で使われる TFIDF 値を領域代数のクエリに拡張した値を利用して関連度計算を行っており、実験ではいくつかのクエリでキーワード検索より精度が低いという結果が出ている。これらの重み付けについてはまだ改善の必要があると考えられる。

参考文献

[1] T. Chinenyanga and N. Kushmerick. Expressive and efficient ranked querying of XML data. In *Proceedings of WebDB-2001*, 2001.

[2] C. L. A. Clarke, G. V. Cormack, and F. J. Burkowski. An algebra for structured text search and a framework for its implementation. *The computer Journal*, 38(1):43–56, 1995.

[3] W. Hersh, C. Buckley, T. J. Leone, and D. Hickam. OHSUMED: an interactive retrieval evaluation and new large test collection for research. In *Proceedings of the 17th International ACM SIGIR Conference*, pages 192–201, 1994.

[4] A. Salminen and F. Tompa. Pat expressions: an algebra for text search. *Acta Linguistica Hungarica*, 41(1-4):277–306, 1994.

[5] A. Theobald and G. Weilkum. Adding relevance to XML. In *Proceedings of WebDB'00*, 2000.