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

増田 勝也† 二宮 崇†‡ 宮尾祐介* 辻井 潤一*‡
†東京大学大学院情報理工学系研究科コンピュータ科学専攻
‡CREST, 科学技術振興機構 *東京大学大学院情報学環
{kmasuda,ninomi,yusuke,tsujii}@is.s.u-tokyo.ac.jp

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すること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 [3, 6, 2, 7, 1]。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持たず、クエリを与えたときにそのクエリの完全一致のみを出力する手法であるかもしくは頑健ではあるが全文書に対して計算を行うために 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。

本研究で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を提案する。頑健性を改善するためにクエリから作られるサブクエリを用いる。これ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文書も検索される。各サブクエリに対して重みを与えることで文書とクエリとの関連度を計算しランキング検索を行う。またスケーラビリティを改善するためにクエリに対する関連度の近似値を用いたフィルタリングを行う。フィルタリングを行うことで関連度計算のコストが削減され、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。

2 背景:領域代数

テキスト中の構造を指定した検索を行う枠組みとして**領域代数** [3, 6] がある。領域代数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。

本研究は [3] 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 ($\rhd$, $\not\rhd$, $\lhd$, $\not\lhd$):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 $r = (p_b, p_e)$(p_b:開始位置、p_e:終了位置) が領域 $r' = (p'_b, p'_e)$ を含む $\Leftrightarrow$ $p_b \leq p'_b \leq p'_e \leq p_e$
この関係を $r' \sqsubset r$ で表す。Clarke らによる領域代数 [3] 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

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

Containing	$G_{q_1 \rhd q_2} = \Gamma(\{a \mid a \in G_{q_1} \wedge \exists b \in G_{q_2}.(b \sqsubset a)\})$
Not Containing	$G_{q_1 \not\rhd q_2} = \Gamma(\{a \mid a \in G_{q_1} \wedge \nexists b \in G_{q_2}.(b \sqsubset a)\})$
Contained In	$G_{q_1 \lhd q_2} = \Gamma(\{a \mid a \in G_{q_1} \wedge \exists b \in G_{q_2}.(a \sqsubset b)\})$
Not Contained In	$G_{q_1 \not\lhd q_2} = \Gamma(\{a \mid a \in G_{q_1} \wedge \nexists b \in G_{q_2}.(a \sqsubset b)\})$
Both Of	$G_{q_1 \triangle q_2} = \Gamma(\{c \mid c \sqsubset (-\infty, \infty) \wedge \exists a \in G_{q_1}.\exists b \in G_{q_2}.(a \sqsubset c \wedge b \sqsubset c)\})$
One Of	$G_{q_1 \triangledown q_2} = \Gamma(\{c \mid c \sqsubset (-\infty, \infty) \wedge \exists a \in G_{q_1}.\exists b \in G_{q_2}.(a \sqsubset c \vee b \sqsubset c)\})$
Followed By	$G_{q_1 \diamond q_2} = \Gamma(\{c \mid c = (p_s, p'_e)$ 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

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。このクエリ '[book] $\rhd$([title] $\rhd$ "retrieval")' は「タイトルに "retrieval" を含む本」を表す。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文書集合全体での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。

本研究ではこの領域代数を基に頑健性及びスケーラビリティを改善したモデルを提案する。

3 頑健性の改善

領域代数を用いた検索ではクエリに対する完全一致のみが検索され、部分一致は検索されない。そこで頑健性を改善するために、与えられた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ことで完全一致だけでなく部分一致も検索されるようにする。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。関連度計算はまずサブクエリと文書の間の関連度を計算し、それらの関連度からクエリ全体とサブクエリの関連度を計算する。

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各ノードを根とする部分木をサブクエリ $q_1, q_2, ..., q_n$ と定義する。

図 1 のクエリ '[book] $\rhd$ ([title] $\rhd$ "retrieval")' の場合、サブクエリは表 2 の $q_1, ..., q_q$ となる。

文書とサブクエリの関連度は、サブクエリのその文書

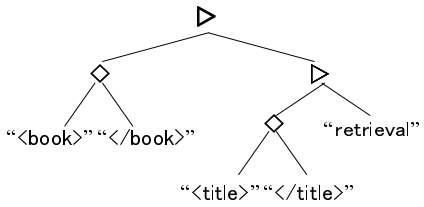

図 1: クエリ ‘[book] ▷([title] ▷ “retrieval”)’ の木構造表現

q_1	“⟨book⟩”
q_2	“⟨/book⟩”
q_3	“⟨title⟩”
q_4	“⟨/title⟩”
q_5	“retrieval”
q_6	‘[title]’
q_7	‘[title] ▷ “retrieval”’
q_8	‘[book]’
q_9	‘[book] ▷ ([title] ▷ “retrieval”)’

表 2: クエリ ‘[book] ▷ ([title] ▷ “retrieval”)’ に対するサブクエリ

での重みと、サブクエリ自身の文書集合での重みから計算される。

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、それらの関連度を用いて以下のように文書とクエリ全体の関連度を計算する。

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。

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

これらの重みや関連度を表す関数についてはさまざまな重みや関数が適用できるが、ここではキーワードに対する重み付けに用いられる TFIDF 値を以下のように構造付クエリに拡張した値を使用する。

定義 4 (クエリの TFIDF 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 値およびクエリ q の IDF 値を以下のように定義する。

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

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

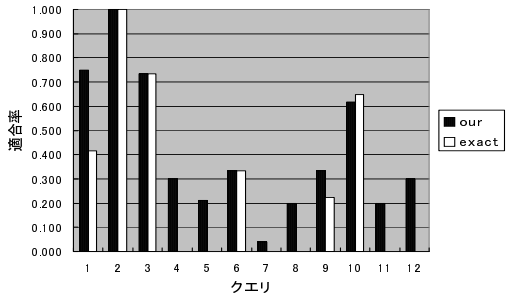

図 2: 適合率 (our 対 $exact$)

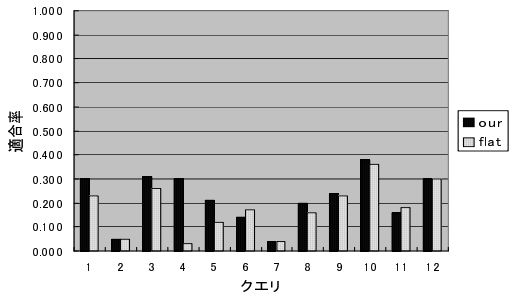

図 3: 適合率 (our 対 $flat$)

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。

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

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

実験

以上のモデルを実装し実験を行った。評価には情報検索の評価に使われる OHSUMED テストコレクション [4] を用いた。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するアブストラクトの集合が与えられている。

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、人手で領域代数のクエリに変換した 12 個のクエリを用いた。使用したクエリの例を表 3 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。

提案モデル (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。提案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

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

表 3: クエリ例

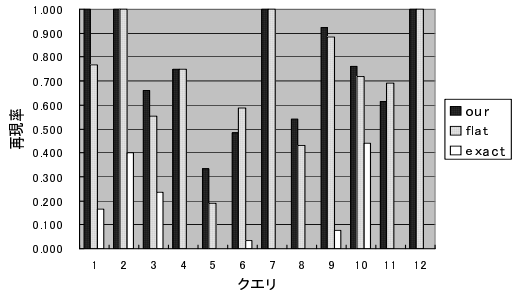

図 4: 再現率

```
function Retrieve(q): ranking list;
    begin
        L := ∅;
        Q := SelectSubquery(q,v);
        Q' := RemoveOverlapQuery(Q);
        S := EvalExact(Q');
        foreach d ∈ S
            begin
                r := CalcRel(d,q);
                PushRanking(L,d,r);
            end
        return L;
    end
```

SelectSubquery: IDF 値が v を超えるサブクエリの集合を返す
RemoveOverlapQuery: Q の中でクエリ q と q' が「q が q' のサブクエリである」という関係にあるクエリ q、q' がある場合 q' の方を取り除き、その結果のクエリ集合を返す
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 に加える

図 5: 検索アルゴリズム

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*) と提案モデル (*our*) を比べると、再現率は提案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ほぼ同じ適合率を示している。またキーワード検索 (*flat*) と提案モデル (*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、再現率ともに提案モデルが上回っている。

4 スケーラビリティの改善：フィルタリング

大規模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。そこで、ある関連度の近似値を定義し、その値が閾値を超える文書に対してのみ関連度計算を行う。閾値を設定することで関連度計算を行う文書数を制限し、高速に関連度計算およびランキングを行うことができる。なおこの関連度の近似値は

- 近似値が高ければ関連度も高い
- 近似値が高い文書を高速に取り出せる

といった特徴を持つことが望ましい。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、フィルタリングで関連度が高い文書を取りこぼすことなく大幅に検索時間を短縮することができる。

このようなフィルタリング手法はキーワード検索においてはあらかじめキーワードに対する重みを計算しておき利用する手法 [5] が提案されている。しかしながら構造付のクエリの場合には、単語と演算子を組み合すことでクエリを無限に考えることができるため、あらかじめクエリに関する重みを計算することはできない。そこで、サブクエリの中でも重みの高いサブクエリを含む文書はクエリとの関連度が高くなると考え、サブクエリの IDF 値で関連度の近似値を以下のように定義する。

$$\phi'(q_i, d) = \begin{cases} idf_q & (\text{文書 } d \text{ がサブクエリ } q_i \text{を含む場合}) \\ 0 & (\text{文書 } d \text{ がサブクエリ } q_i \text{を含まない場合}) \end{cases}$$

$$\Phi'(q, d) = \max_i \phi'(q_i, d)$$

フィルタリングを用いた検索アルゴリズムを図 5 に示す。クエリが与えられた後、クエリからサブクエリを作る。次に各サブクエリに対して IDF 値を無作為抽出された文書集合において計算し、IDF 値が閾値を超えるサブクエリの集合を作る。その集合の中で、クエリ q と q' が「クエリ q が q' のサブクエリである」という関係にあるクエリ q、q' がある場合には、q' の方を取り除く。その結果のクエリ集合中のクエリを含む文書を全て取り出し、それらの文書に対してのみスコア計算を行い、ランキングを作成する。

サブクエリの IDF 値の近似計算

IDF 値の計算は無作為抽出された文書集合中で近似的に計算する。正確な IDF 値を計算するためには全ての文書に対して各サブクエリの有無を計算しなければならず、時間がかかってしまう。そこで無作為抽出を用いて高速に近似的に IDF 値を計算する。

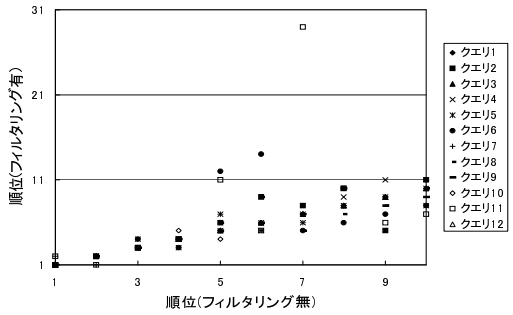

図 6: フィルタリングによる上位 10 文書の順位変化

関連度計算を行う文書集合の取得

各サブクエリの IDF 値が計算された後、以下で表されるサブクエリを含む文書集合を取り出す。

$$\{q_i | idf_{q_i} > v \wedge \nexists q_j.(idf_{q_j} > v \wedge q_j \prec q_i)\}$$

(v:閾値、$q_j \prec q_i$:q_j が q_i のサブクエリである) 例えばクエリ '[title] ▷ "retrieval" ' の IDF 値が閾値を超えた場合、そのサブクエリ '[title]'、"retrieval" の IDF 値が閾値を超えなければ '[title] ▷ "retrieval" ' を用いる。もし "retrieval" の IDF 値も閾値を超えていれば '[title] ▷ "retrieval" ' は用いずに "retrieval" のみを用いる。なぜならこの場合、'[title] ▷ "retrieval" ' を含む文書は必ず "retrieval" を含むため、'[title] ▷ "retrieval" ' を用いる必要はないからである。

実験

3 章の実験で用いた OHSUMED テストコレクションを用いてフィルタリングの実験を行った。無作為抽出で 5000 文書を取り出し IDF 値の計算を行った。また閾値としては $4.605 = \log \frac{5000}{50}$ を用いた。すなわちフィルタリングには無作為抽出された 5000 文書中、50 文書以下で出現するサブクエリを用いた。実験はフィルタリングの有無による検索時間の違いおよび検索結果の文書の順位の変化を調べた。

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、検索時間が大幅に短縮されていることがわかる。

また、フィルタリング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 値の変化によって順位が多少変化しているものの、フィルタリング無しでの上位の文書がフィルタリングにおいて選び出されないということはなかった。

5 まとめと今後の課題

本論文で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を提案した。クエリから作られる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、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。この結果の原因としては、実験に用いた OHSUMED テストコレクションにはもともと簡単な文書構造以外の構造はついておらず、クエリも自然言語のクエリであり人手で構造をつけたため構造を指定することが検索結果にそれほど良い影響を与えない可能性がある、という点も考えられるが、重み付けについてはまだ改善の余地があると考えられる。またフィルタリングに関しては、今回はサブクエリの IDF 値を用いているがその計算に無作為抽出を用いているためその誤差による IDF 値の変化が順位の変化をもたらしており、IDF 値の計算法に改善の必要があると考えられる。

参考文献

[1] D. Carmel, Y. S. Maarek, M. Mandelbrod, Y. Mass, and A. Soffer. Searching XML documents via XML fragments. In *Proceedings of the 26th International ACM SIGIR Conference*, pages 151–158, 2003.

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

[3] 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.

[4] 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.

[5] M. Persin, J. Zobel, and R. Sacks-Davis. Filtered document retrieval with frequency-sorted indexes. *Journal of the American Society of Information Science*, 47(10):749–764, 1996.

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

[7] T. Schlieder and H. Meuss. Querying and ranking XML documents. *JASIST*, 53(6):489–503, 2002.