社団法人 情報処理学会　研究報告
IPSJ SIG Technical Report

2004－NL－160　(17)
2004／3／5

質問応答システムにおける逓減加点法に基づく複数記事情報の利用

村田 真樹　　内山 将夫　　井佐原 均

独立行政法人 通信総合研究所

〒619-0289 京都府相楽郡精華町光台 3-5

TEL:0774-98-6833　FAX:0774-98-6961　{murata,mutiyama,isahara}@crl.go.jp

あらまし

質問応答システムの精度向上を目的として複数の記事の情報を得点を減らしながら利用する新しい方法を提案する．しばしば質問の解は複数の記事で発見される．そのような場合は一つの記事に基づくよりは複数の記事を使って解の推定を行なった方がより良い解を得ることができるだろう．そこで，我々の提案手法では，複数の記事から得られた解の候補の得点を加算することで複数の記事の情報を利用する．しかし，単純に得点を加算したのでは逆にシステムの性能を下げることになる．そこで，我々の方法では，この単純に加算する欠点を減らすために，得点を少しずつ減らしながら加算する．QAC のテストコレクションを用いて実験を行ない，統計的検定を用いて我々の手法が有意に優れていることを確認する．提案手法による精度向上は 0.05 から 0.14 とかなり大きい．これらの結果に加え，我々の手法はたいへん簡便でなおかつ使いやすいものなので，質問応答システムにおいて大いに役に立つものとなると思われる．

キーワード　　質問応答，複数記事，逓減加点，融合法

Multiple Documents as Evidence with Decreased Adding in a Question-answering System

Masaki Murata　　Masao Utiyama　　Hitoshi Isahara

Communications Research Laboratory

3-5 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0289, Japan

TEL:+81-774-98-6833　FAX:+81-774-98-6961　{murata,mutiyama,isahara}@crl.go.jp

Abstract

We propose a new method of using multiple documents as evidence with decreased adding to improve the performance of a question-answering system. Sometimes, the answer to a question may be found in multiple documents. In such cases, using multiple documents for prediction would generate better answers than using a single document. Thus, our method employs information from multiple documents by adding the scores of the candidate answers extracted from the various documents. Because simply adding scores degrades the performance of question-answering systems, we add scores with decreasing weights to reduce the negative effect of simple adding. We carried out experiments using the QAC test collection and confirmed the effectiveness of our method through a statistical test. The degree of improvement by our method was quite large, ranging from 0.05 to 0.14. These results, and the fact that our method is very simple and easy to use, demonstrate its feasibility and utility for question-answering systems.

key words　　Question Answering, Multiple Documents, Decreased Adding, Combined Method

1　はじめに

質問応答システムとは，与えられた質問に対してその答えを出力するシステムのことで，例えば，「日本の首都はどこですか」という質問文が与えられると，「東京は日本の首都で，その国の最も大きく重要な都市であり，東京は日本の４７都道府県のうちの一つである．」という文をウェブや新聞記事などの電子テキストから探し出し，「東京」と答える．質問応答システムは情報検索の代りとして重要になるだろうし，また将来の人工知能システムの基本要素にもなるであろう重要なものである[1, 2, 3]．近年，多くの研究者がこの重要な質問応答システムの研究をしている[4, 5, 6, 7, 8]．質問応答システムに関する評価型ワークショップも開かれるようになり，米国では TREC[9] が，日本では NTCIR の QAC[10] がある．これらの評価型ワークショップは，複数のチームの性能を同一の基準で比較することで技術の発展に貢献している．われわれは，早くから質問応答システムを研究し[11]，また， QAC[12] にも参加しながら質問応答システムの改良を続けてきた[2]．

この論文では，我々は質問応答システムの精度向上のために，得点を減らしながら複数の記事 (文書) の得点を利用する新しい方法を提案する[1]．本稿ではこの方法を逓減加点法と呼ぶ．質問の答えが複数の記事で見つかることは多く，そのような場合は，複数の記事を使って答えを推定した方が一つの記事を使って推定するよりも良い答えを得ることができるだろう．我々の方法では，複数の記事から得た解の候補の得点を加算することで，複数の記事の情報を利用する．しかし，ただ単純に得点を加算するだけではシステムの性能を下げる場合がある．そこで，我々の方法では，この単純に加算する際に生じる問題に対処するために，得点の加算の際に得点を減らしながら加算する手法を用いる．より具体的に言うと，我々の方法では，i 番目の解の候補の得点には $k^{(i-1)}$ の重みをかけておいてその後で得点を加算する．最終的な答えは合計得点により判断する．例えば，「東京」が三つの記事から解の候補として抽出され，それらの得点が 26， 21， 20 であり，k が 0.3 であったとしよう．この場合，「東京」の合計得点は 34.1 となる ($= 26 + 21 \times 0.3 + 20 \times 0.3^2$)．このような方法でそれぞれの候補の得点を計算し，最も高い合計得点を持つ候補を解とする．

この我々の方法を評価するために，我々は， QAC のテストコレクション[12] を利用し，また評価では有意差検定を用いた．この実験では，五つの種類の質問応答システムを利用してそれらでこの方法が有効かどうかを調べた．また，単純に記事の得点を加算するだけの方法ではシステムの性能を下げる場合があることも確認した．我々の方法はシステムの性能を向上させることのできるものであり，それに加えて単純で使いやすい特徴があるもので，非常に優れた方法である．

表 1: 解の候補とオリジナルの得点 (「東京」が正解である場合)

順位	解の候補	得点	記事番号
1	京都	3.3	926324
2	東京	3.2	259312
3	東京	2.8	451245
4	東京	2.5	371922
5	東京	2.4	221328
6	北京	2.3	113127
…	…	…	…

表 2: 解の候補と単純に加算された得点 (「東京」が正解である場合)

順位	解の候補	得点	記事番号
1	東京	10.9	259312, 451245, 371922, 221328
2	京都	3.3	926324
3	北京	2.3	113127
…	…	…	…

2　複数記事の利用における逓減加点法の利用

「日本の首都はどこですか」という質問文が与えられたとしよう．このとき，得るべき答えは「東京」である．一般的な質問応答システムは，表 1のように，解の候補と得点をリストとして出力でき，また，解の候補を取り出した記事を指し示す記事番号も出力することができる．

表 1の例だと，最も得点の大きい候補は「京都」であり，誤った解を出力することになる．

解の候補の得点を単純に加算する方法は，すでに提案されている[13][2]．さきほどの例だと，この方法は表 2の結果を得て，正しく「東京」を解として出力することができる．この方法は，複数の記事の情報を利用することで正しい解を得ることができた．

しかし，この方法には，高頻度の解の候補を取り出しやすいという問題がある．これは，特に性能が高いシステムで深刻な問題で，もともと性能が高いシステムで

[1] 本稿で提案する逓減加点法に基づく質問応答システムは特許出願中である．また，本稿では i 番目の解の候補の得点に $k^{(i-1)}$ の重みをかける方法を用いているが，i 番目の解の候補の得点にかける重みを a^i とし，a^i の値を一つ一つ重回帰分析などでもとめるようなことをしてもよく，重みの与え方には様々な方法がありうるだろう．

[2] 他に先行研究としては複数記事から得た解の候補の頻度をその解の候補のスコアとして利用する方法[14] がある．

表 3: 解の候補とオリジナルの得点 (「京都」が正解の場合)

順位	解の候補	得点	記事番号
1	京都	5.4	926324
2	東京	2.1	259312
3	東京	1.8	451245
4	東京	1.5	371922
5	東京	1.4	221328
6	北京	1.3	113127
…	…	…	…

表 4: 解の候補と単純に加算した得点 (「京都」が正解の場合)

順位	解の候補	得点	記事番号
1	東京	6.8	259312, 451245, 371922, 221328
2	京都	5.4	926324
3	北京	1.3	113127
…	…	…	…

表 5: 解の候補と逓減加点法に基づく得点 (「京都」が正解の場合)

順位	解の候補	得点	記事番号
1	京都	5.4	926324
2	東京	2.8	259312, 451245, 371922, 221328
3	北京	1.3	113127
…	…	…	…

は，システムの出力した元の得点の方が単純に加算した得点よりも信頼できる場合が多く，単純に加算する方法はしばしばシステムの性能を劣化させることになる．

この問題に対処するために，我々は得点を減らしながら加算する新しい方法を開発した．解の候補の得点を単純に加算する代りに，得点を減らす重みをつけて得点を加算するのである．この方法は，高頻度語を取り出しやすいという悪い効果を減じ，なおかつシステムの性能を向上させる効果を持つ．

この方法の有効性を示す例をあげる．「日本の首都は西暦 1000 年の時はどこでしたか．」と質問が与えられ，システムは表 3の出力をしたとしよう．正解は「京都」であり，もともとこのシステムは正解を出力している．

しかし，単純に加算する方法を用いるとその結果は表 4のようになり，間違った解の「東京」をシステムの解としてしまう．

ここで得点を減らしながら加算する我々の新しい方法

表 6: 解の候補と逓減加点法に基づく得点 (「東京」が正解の場合)

順位	解の候補	得点	記事番号
1	東京	4.3	259312, 451245, 371922, 221328
2	京都	3.3	926324
3	北京	2.3	113127
…	…	…	…

を利用してみよう．ここでは，細かいシステムの仕様として，i 番目の候補の得点に $0.3^{(i-1)}$ を乗じることとする．その場合，「東京」の得点は 2.8 であり ($= 2.1 + 1.8 \times 0.3 + 1.5 \times 0.3^2 + 1.4 \times 0.3^3$)，システムの結果は表 5のようになり，「京都」の方がまだ点が高いので正解の「京都」を解として正しく出力することができる．

我々の新しい方法は最初の例 (「日本の首都はどこですか」) でも正しい解を得ることができる．最初の例に適用すると，「東京」の得点は 4.3 となり ($= 3.2 + 2.8 \times 0.3 + 2.5 \times 0.3^2 + 2.4 \times 0.3^3$)，結果は表 6のようになり，「東京」が最も高い得点となり解として正しく出力される．

以上述べたように，得点を減らしながら加算する我々の方法は，例にあげたどの例も正しく解を求めることができるものである．我々の方法は，高頻度の解の候補を取り出しやすい欠点を減じながら，なおかつ複数記事の情報を利用し精度向上を実現できるものである．

3 本研究で用いる質問応答システム

本節では本研究の実験で用いる質問応答システムについて述べる．詳細は文献[(2)] を参照してほしい．このシステムは以下の三つの基本要素からなる．

1. 解表現の推定
   システムは疑問代名詞の表現などに基づいて解表現 (解がどのような言語表現か) を推定する．例えば，入力の質問文が「日本の面積はどのくらいですか」だとする (図 1) と「どのくらい」という表現から解表現は数値表現であろうと推測する．
2. 文書検索
   システムは質問文からキーワードを取り出し，これらのキーワードを用いて文書を検索する．この検索により，解が書いてありそうな文書群を集めることになる．例えば，入力の質問文が「日本の面積はどのくらいですか」だとすると，「日本」「面積」がキーワードとして抽出され，これらを含む文書を検索することになる．
3. 解の抽出
   システムは解が書いてありそうな文書群から，推定した解表現に適合する言語表現を抽出し，それを解とし

て出力する．例えば，入力の質問文が「日本の面積はどのくらいですか」だとすると，文書検索で検索した「日本」「面積」を含む文書群から，解表現として推定した数値表現にあたる言語表現を解として抽出する．

以降の節で，それぞれをより詳しく説明する．

3.1 解表現の推定

人手で作成したヒューリスティックルールを使って解表現を推定する． 39 個のルールを作成した．そのいくつかを以下に示す．

- 質問文に「誰」という表現がある場合，解表現は人名である．
- 質問文に「いつ」という表現がある場合，解表現は時間表現である．
- 質問文に「どのくらいの」という表現がある場合，解表現は数値表現である．

3.2 文書検索

文書検索のためのキーワードは ChaSen[15] により取り出し付属語などはキーワードから除外した．

文書検索は以下のように行なう．

1. まず以下の式で文書検索を行ない，上位 k_{dr1} 個の記事を取り出す．

$$Score(d) = \sum_{\text{term } t} \left(\frac{tf(d,t)}{tf(d,t) + k_t \frac{length(d) + k_+}{\Delta + k_+}} \times log \frac{N}{df(t)} \right) \quad (1)$$

ただし， d は記事で t は質問文から取り出したキーワードで， $tf(d,t)$ は，記事 d に出現するキーワード t の頻度で， $df(t)$ はキーワード t が出現する頻度で， N は記事の総数で， $length(d)$ は記事 d の長さで， Δ は記事長の平均である． k_t と k_+ は実験で定める定数である．この式はロバートソンの Okapi ウェイティング[16, 17] の式に基づくもので，我々も情報検索でよく用いる式である[18, 19]．ただし，質問応答では多くの種類のキーワードがマッチすることが重要なので k_t の値としては大きな値を用いる．

2. 次に，以下の式で記事をリランキングし，上位 k_{dr2} 個の記事を取り出す．

$$Score(d) = max_{t1 \in T} \sum_{t2 \in T3} w_{dr2}(t2) log \frac{N}{2dist(t1,t2) * df(t2)} \quad (2)$$

$$T3 = \{t | t \in T, 2dist(t1,t) \frac{df(t)}{N} \leq 1\}, \quad (3)$$

ただし， T はキーワードの集合で， $dist(t1,t2)$ はキーワード $t1$ と $t2$ の間の距離で便宜上 $t1 = t2$ のとき $dist(t1,t2) = 0.5$ としている． $w_{dr2}(t2)$ は $t2$ の関数で実験により定められる．

一般には質問応答システムでは質問文から取り出した複数のキーワードが近くに出現することを保証するために記事を段落などの小さい単位に分割するが，本システムでは式 2のリランキングによりキーワードが近くにある場合に得点をあげる式を用いるので，記事を分割する必要がなく記事をそのまま文書検索に使えるのである．この文書検索では上位 20 記事を取り出し，それを次の解の抽出で利用する．

3.3 解の抽出

文書検索で得た記事から，名詞，未知語連続を取り出しそれらを解の候補とする．それぞれの候補には，解の候補とキーワードの近さに基づく得点 $Score_{near}(c)$ と解表現の意味制約を満足しているかいなかに基づく $Score_{sem}(c)$ の二つの得点を与え，その合計点が最も大きい候補を解とする．

$Score_{near}(c)$ は以下の式で与えられる．

$$Score_{near}(c) = \sum_{t2 \in T3} w_{dr2}(t2) log \frac{N}{2dist(c,t2) * df(t2)} \quad (4)$$

$$T3 = \{t | t \in T, 2dist(c,t) \frac{df(t)}{N} \leq 1\} \quad (5)$$

ただし， c は解の候補であり， $w_{dr2}(t2)$ は実験で定められる関数である．

解表現の意味制約に基づく得点 $Score_{sem}(c)$ は，人手で作成した規則により与えられる． 45 の規則を作成した．そのいくつかを以下に示す．

1. 推定した解表現 (人名や地名など) と一致する候補に 1000 を与える．解の候補が人名か地名かと特定する方法には SVM に基づく固有表現抽出技術を利用した．
2. 解表現が「国名」の場合に解の候補が国名のときに 1000 を与える．
3. 質問文が「何 + 名詞 X」の場合，名詞 X を最後に持つ候補に 1000 を与える．

4 実験

4.1 実験で用いるデータ

実験には NTCIR3 の QAC1 の Task-1,2 のデータを使った．このデータはそれぞれ 200 問の質問とその答えの対で構成されている．また， CD 毎日新聞 1998,1999 年版の記事を使って作成されている． Task-1 では 5 つの解を出力する． Task-2 では複数の解が正解になる質問も与えられ，すべての解を出力すると満点になる．例えば，「国連の常任理事国はどこですか」という質問に対してすべての常任理事国を解として出力するとその問題は満点となる． Task-1 では MRR が Task-2 では MF が評価に用いられる． MRR は r 個目に正解を出力すると $1/r$ の得点が与えられる． MF は F-measure の平均のこ

とで各質問ごとの F-measure を全問題で平均した値のことである．

我々のシステムでは Task-1 では上位 5 つを出力した．Task-2 では最初の解一つのみを出力した．これは，我々のシステムは Task-2 のように複数の解を答える機能がないため，再現率よりも適合率を重視してそのようにした．

4.2 得点加算法

実験では，以下の得点加算法を利用した．

- オリジナル法
  元のシステムの出力のまま．(得点の加算を行なわない)
- 単純加算法
  複数の記事から取り出した解の候補の得点を加算し，その得点をそのまま加算した合計得点に基づき解を出力する．
  我々の質問応答システムでは，$Score_{sem}(c)$ を計算するヒューリスティックルールで 1000 の倍数の得点を多く与えるので，解の候補の重要性は 1000 を単位に大きく変わる．そこで，得点の加算のときに 1000 よりも上の桁の数字を変更したくない．そこで，1000 よりも下の桁の数字を取り出し，その数字をすべて加算して，その結果を元の 1000 よりも上の数字とくっつけてそれを合計した後の得点とする．ただし，1000 よりも上の数字が異なるものは加算しない．
  例えば，ある解の候補が二回現れそれらが 1025 と 1016 であったとする．そのときは，25 と 16 が取り出され，その和の 41 と 1000 をくっつけて 1041 となる．また，ある解の候補が二回現れそれらが 2025 と 2016 とすると 2041 となる．また，ある解の候補が二回現れそれらが 2025 と 1016 とすると 2025 となる．
- 逓減加点法
  単純加算法と同じく，複数記事から取り出した候補の得点を加算する．1000 の桁を使って得点を分割して加算するのも単純加算法と同じである．
  しかし，値の加算方法は異なる．この方法は i 番目の候補の得点には $k^{(i-1)}$ の値を乗じてから得点を加算する．以下の式で表される．

$$Score_{decreased} = \sum_{1 \le i \le n} k^{i-1}\ score_{original}(i) \qquad (6)$$

  ただし，$Score_{decreased}$ は，最終的な加算後の値の 1000 より下の桁の数字で，$score_{original}(i)$ は，元の値の 1000 より下の桁の数字である．n は 1000 より上の桁で同じ数字を持つ複数の記事から得られた同じ解の候補の出現回数である．k は実験で定める定数である．
  例えば，ある解の候補が二回出現し，それらが 2025 と 2016 で $k = 0.3$ とすると，25 と 16 が取り出され式 6の計算により $29.8(= 25 + 16 \times 0.3)$ を得て元の 2000 とくっつけて 2029.8 が最終的な値となる．
  この研究では，k としては 0.01, 0.02, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 の 12 種類を使った．
- 融合法
  この方法は，オリジナル法，単純加算法，逓減加算法 (12 種類の k) の組み合わせである．この方法はまず学習データでこれらの方法のうちどの方法が最も良い精度を出すかを調べて，最も精度の高かった方法を利用して問題を解く．
  本研究では QAC1 のデータ以外を用いないため学習データがないので，学習のために 10 分割のクロスバリデーションを利用した．
  この方法には二つの効果がある．一つは，融合による精度向上である．もう一つは公平な評価のためである．本研究では多くの種類の k を用いるために，ある k で偶然良い精度を出すかもしれなく，そのときの精度をもって本システムの精度ということは難しい．これに対して融合法ではクロスバリデーションを使うので公正な精度を求めることができる．

4.3 システムの種類

本稿の手法が複数のシステムで有効であることを確認するために，実験には以下の五つのシステムを利用した．システム 5 は 3節で述べたシステムそのもので，他の四つのシステムはシステム 5 を少し変更して作成した．

- システム 1(Sys1)
  このシステムはシステム 5 において $Score_{sem}(c)$ を使わないものである．すなわち，解表現に関するヒューリスティックルールに基づく得点を一切使わない．
- システム 2(Sys2)
  このシステムはシステム 5 において式 4の代りに以下の式 7を使うものである．

$$Score_{near2}(c) \quad = \quad \sum_{t2 \in T} w_{dr2}(t2) log \frac{N}{df(t2)} \qquad (7)$$

  すなわち，システム 2 は，質問文から取り出したキーワードと解の候補の距離の情報を使わないものである．
- システム 3(Sys3)
  このシステムはシステム 2 において，文書検索で記事を段落に分割するシステムで，式 2での記事のリランキングを行なわないものである．
  システム 3 はシステム 2 と同じく質問文から取り出したキーワードと解の候補の距離の情報を使わないが，

表 7: Task-1 での結果 (MRR)

	Sys1	Sys2	Sys3	Sys4	Sys5
オリジナル法	0.130	0.294	0.405	0.526	0.541
単純加算法	0.051^{--}	0.387^{++}	0.474^{+}	0.523	0.449^{--}
融合法	0.097	0.437^{++}	0.506^{++}	0.567^{+}	0.597^{++}
逓減加点法					
k=0.01	0.115	0.432^{++}	0.498^{++}	0.544	0.551
k=0.02	0.115	0.433^{++}	0.502^{++}	0.551^{+}	0.561
k=0.05	0.115	0.440^{++}	0.510^{++}	0.549	0.563
k=0.1	0.109	0.449^{++}	0.516^{++}	0.556	0.570
k=0.2	0.129	0.446^{++}	0.509^{++}	0.574^{++}	0.590^{++}
k=0.3	0.130	0.450^{++}	0.504^{++}	0.571^{+}	0.597^{++}
k=0.4	0.111	0.434^{++}	0.504^{++}	0.567^{+}	0.580
k=0.5	0.101	0.428^{++}	0.509^{++}	0.565	0.565
k=0.6	0.100	0.414^{++}	0.505^{++}	0.544	0.544
k=0.7	0.081	0.411^{++}	0.498^{++}	0.531	0.537
k=0.8	0.072^{-}	0.399^{++}	0.489^{++}	0.528	0.492
k=0.9	0.062^{--}	0.390^{++}	0.480^{++}	0.532	0.472^{-}

表 8: Task-2 での結果 (MF)

	Sys1	Sys2	Sys3	Sys4	Sys5
オリジナル法	0.065	0.172	0.246	0.359	0.376
単純加算法	0.038	0.270^{++}	0.324^{+}	0.374	0.317
融合法	0.045	0.311^{++}	0.362^{++}	0.412^{+}	0.451^{++}
逓減加点法					
k=0.01	0.060	0.291^{++}	0.360^{++}	0.381	0.381
k=0.02	0.064	0.291^{++}	0.360^{++}	0.388^{+}	0.393
k=0.05	0.059	0.296^{++}	0.365^{++}	0.395^{+}	0.399
k=0.1	0.051	0.311^{++}	0.365^{++}	0.402^{+}	0.416^{+}
k=0.2	0.066	0.321^{++}	0.361^{++}	0.435^{++}	0.446^{++}
k=0.3	0.066	0.318^{++}	0.357^{++}	0.432^{++}	0.451^{++}
k=0.4	0.044	0.305^{++}	0.362^{++}	0.427^{++}	0.421
k=0.5	0.039	0.300^{++}	0.372^{++}	0.431^{+}	0.421
k=0.6	0.049	0.285^{++}	0.372^{++}	0.405	0.401
k=0.7	0.033	0.285^{++}	0.362^{++}	0.388	0.399
k=0.8	0.035	0.277^{++}	0.350^{++}	0.383	0.351
k=0.9	0.042	0.270^{++}	0.339^{++}	0.385	0.336

記事を段落に分割しているので，同じ段落内にキーワードと解の候補が含まれることは保証しており，システム 2 よりは距離の情報を少々は用いているともいえるものである．

- システム 4(Sys4)
  このシステムはシステム 5 において文書検索で記事を段落に分割するシステムであり，式 2での記事のリランキングを行なわないものである．
  現在最も一般的な質問応答システムは，記事を段落に分割して文書検索を行なうものなので，システム 4 は最も一般的な質問応答システムに相当する．
- システム 5(Sys5)
  システム 5 は 3節で述べた本研究の実験で基本とする質問応答システムである．

4.4 実験結果

以上の節で述べた方法で実験を行なった．結果を表 7と表 8に示す． t 検定を行ないオリジナル法を基準とし，それよりも 5% の有意水準で精度向上しているものに “+” を 1% のものに “++” をつけた．また 5% の有意水準で精度劣化しているものに “−” を 1% のものに “−−” をつけた．

5 考察

5.1 実験結果の整理

表 7と 8から，以下のことがわかる．

- 単純加算法はシステム 2,3 などの場合でオリジナル法よりも精度が上がるが，システム 5 でオリジナル法よりも精度が下がる．
- 我々の提案手法の融合法は，システム 1 以外の多くのシステムでオリジナル法よりも高い精度をあげた．
- 融合法は常に単純加算法よりも精度が高い．
- 提案手法は 0.05 から 0.14 の精度向上をあげた．
- 逓減加点法では k として 0.2 と 0.3 がよい． (実際に融合法でどの k が選ばれる回数が多いかを調べたが， 0.2 と 0.3 の時が多かった．)
- 我々の最良のシステム 5 では，融合法は 0.597 と 0.451 の値を Task-1， Task-2 のそれぞれで得た．オリジナル法のそれは 0.541 と 0.376 であった． Task-1 の 1 位の精度が 0.608 で Task-2 の 1 位の精度が 0.364 であるので，我々の最良のシステム 5 で融合法を利用すると， Task-1 の精度は 1 位の精度にかなり近く，Task-2 の精度は 1 位の精度よりもはるかに高く，非常に高性能な結果となる．

5.2 解を含む記事の数と精度の関係

複数記事の情報を得点を減らしながら用いる提案手法には以下の問題がある．解を含む記事が一つだけの場合，複数の記事の解の得点を加算できずシステムの性能を逆に下げてしまう．最も性能の良いシステム 5 を用いてこの問題を調べてみた．文書検索で得られた上位 20 記事のうちの何記事 (x 記事) に解が含まれるかによって精度がどうなるかを計算した．これを表 9と表 10に示す．複数の正解を持つ質問の場合は， x はそれぞれの正解を含む記事の数の平均とした．この結果も t 検定を行なった．表記方法は表 7と表 8と同じである．五つの質問で上位 20 記事で解を含む記事が得られず，それらの場合は精度はすべて 0 となっている．

我々の予想どおり， $0<x\leq 1$ の場合に融合法はオリジナル法よりも精度が低かった．しかし，それ以外の場合は融合法の方が精度が高かった．単純加算法は， Task-1

表 9: 解を含む記事数が x である場合のシステム 5 の Task-1 の結果 (MRR)

	$x=0$	$0<x\leq 1$	$1<x\leq 2$	$2<x\leq 3$	$3<x\leq 5$	$5<x\leq 10$	$x>10$
質問文の数	5	30	33	36	28	43	25
オリジナル法	0.000	0.493	0.412	0.694	0.529	0.563	0.632
単純加算法	0.000	0.267^{--}	0.170^{--}	0.428^{--}	0.514	0.651	0.728
融合法	0.000	0.427	0.424	0.744	0.636^{+}	0.670^{+}	0.776
逓減加点法							
k=0.01	0.000	0.493	0.412	0.700	0.529	0.558	0.712
k=0.02	0.000	0.493	0.412	0.717	0.564	0.567	0.712
k=0.05	0.000	0.480	0.412	0.706	0.557	0.586	0.736
k=0.1	0.000	0.453	0.436	0.700	0.564	0.609	0.752
k=0.2	0.000	0.447	0.418	0.733	0.629	0.651^{+}	0.752
k=0.3	0.000	0.427	0.424	0.744	0.636^{+}	0.670^{+}	0.776
k=0.4	0.000	0.380^{-}	0.370	0.728	0.614	0.679^{+}	0.792^{+}
k=0.5	0.000	0.360^{-}	0.333^{-}	0.656	0.621	0.712^{+}	0.792^{+}
k=0.6	0.000	0.353^{-}	0.291^{--}	0.572	0.614	0.721^{+}	0.792
k=0.7	0.000	0.293^{--}	0.285^{--}	0.567^{-}	0.614	0.740^{++}	0.792
k=0.8	0.000	0.287^{--}	0.248^{--}	0.489	0.536	0.679	0.792
k=0.9	0.000	0.280^{--}	0.212^{--}	0.461^{--}	0.521	0.670	0.752

の 0<x≤5 の場合と Task-2 の 0<x≤3 の場合でオリジナル法よりも精度が低かった．単純加算法の方が融合法よりも多くの場合で，オリジナル法よりも精度が低く，この結果も融合法の方が単純加算法よりも優れていることを示す．

0<x≤1 の場合の融合法の欠点を対処しさらにシステムの精度向上を図るには以下の方法が有望である．

- 記事セットの規模を大きくする．
  結果からわかるように融合法のシステムの性能は解を含む記事の数が大きくなるにつれて向上する．このため，ウェブのテキストのようにより大きな記事セットを用いると，解を含む記事の数も大きくなり精度も向上すると思われる．
- 解を含む記事の数が一個かそれとも二個以上であるのかを同定する．
  0<x≤1 の場合は融合法はオリジナル法よりも性能が低い．このため，もし，解を含む記事の数が一個かそれとも二個以上であるのかを同定することができると，記事の数が一個の場合にオリジナル法を使いそれ以外の場合に融合法を用いることができ，精度を向上させることができる．

6 おわりに

我々は，質問応答システムは文書検索の代りのより便利な検索システムとして利用され，また，将来の人工知能システムの基本要素として利用され，より重要なものとなると考えている．そこで我々は質問応答システムの精度向上を目的として，得点を減らしながら複数記事の情報を利用する新しい方法を提案した．解が複数の記事に現れることがあり，そのような場合は一つだけの記事から解の推定を行なうよりも，複数の記事を利用して解の推定を行なった方がより確からしい解を抽出することができる．我々の方法では複数の記事から取り出された同じ解の候補の得点を加算することで複数の記事の情報を利用する．しかし，単純に得点を加算する方法だと逆にシステムの性能を下げる場合がある．そこで，我々の方法では得点を小さくしながら加算することで，単純に得点を加算する方法の欠点を補う．我々は QAC のテストコレクションを使って実験をし，統計的検定を利用して我々の手法が有意に有効であることを確認した．また，単純に得点を加算する方法だといくつかの場合で質問応答システムの性能を下げる場合があることも確認した．得点を減らしながら加算する提案手法では，i 番目の解の候補には加算する前に $k^{(i-1)}$ の値を乗じる．我々の実験ではこの k としては 0.2 や 0.3 が良いことがわかった．我々の提案する方法は，大きい精度向上を生むものであり，それに加えて単純で使いやすく，これらの結果および特徴は提案手法の優秀性を物語る．我々はさらにいくつかの種類の質問応答システムでも実験し，一つのシステムを除くすべての場合で我々の手法が有効であることを確認した．

将来の研究としては，我々はウェブのテキストを使うなどして文書の記事セットの規模を大きくしたいと思っている．また，解を含む記事の数が一つかそれとも二つ以上かの同定を行ないさらに提案手法の性能を向上させたいと思っている．このような研究を続けて，質問応答システムの性能を向上させ，いつかは人工知能システムの一部として質問応答システムを利用できるようにしたいと思っている．

表 10: 解を含む記事数が x である場合のシステム 5 の Task-2 の結果 (MF)

	$x=0$	$0<x\leq 1$	$1<x\leq 2$	$2<x\leq 3$	$3<x\leq 5$	$5<x\leq 10$	$x>10$
質問文の数	5	30	33	36	28	43	25
オリジナル法	0.000	0.413	0.285	0.522	0.293	0.353	0.456
単純加算法	0.000	0.187^{--}	0.030^{--}	0.278^{-}	0.336	0.526^{+}	0.592
融合法	0.000	0.353	0.333	0.583	0.329	0.512^{++}	0.656^{+}
逓減加点法							
k=0.01	0.000	0.413	0.285	0.522	0.257	0.353	0.536
k=0.02	0.000	0.413	0.285	0.539	0.293	0.367	0.536
k=0.05	0.000	0.413	0.285	0.528	0.293	0.386	0.576
k=0.1	0.000	0.387	0.345	0.528	0.293	0.405	0.616^{+}
k=0.2	0.000	0.387	0.333	0.556	0.364	0.488^{+}	0.616^{+}
k=0.3	0.000	0.353	0.333	0.583	0.329	0.512^{++}	0.656^{+}
k=0.4	0.000	0.287^{-}	0.242	0.539	0.336	0.512^{+}	0.672^{+}
k=0.5	0.000	0.287^{-}	0.212	0.467	0.371	0.572^{++}	0.672^{+}
k=0.6	0.000	0.287^{-}	0.182	0.361	0.371	0.591^{++}	0.672^{+}
k=0.7	0.000	0.187^{--}	0.182	0.389	0.371	0.628^{++}	0.672^{+}
k=0.8	0.000	0.187^{--}	0.121^{-}	0.306^{-}	0.336	0.544^{+}	0.672^{+}
k=0.9	0.000	0.187^{--}	0.061^{--}	0.306^{-}	0.336	0.544^{+}	0.632^{+}

参考文献

(1) 村田真樹, 内山将夫, 井佐原均, 類似度に基づく推論を用いた質問応答システム, 自然言語処理研究会 2000-NL-135, (2000), pp. 181–188.

(2) Masaki Murata, Masao Utiyama, and Hitoshi Isahara, A question-answering system using unit estimation and probabilistic near-terms ir, *Proceedings of the Third NTCIR Workshop (QAC)*, (2002).

(3) 村田真樹, 質問応答システムの現状と展望, 電子情報通信学会学会誌, Vol. 86, No. 12, (2003), pp. 959–963.

(4) Julian Kupiec, MURAX: A robust linguistic approach for question answering using an on-line encyclopedia, *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, (1993).

(5) Abraham Ittycheriah, Martin Franz, Wei-Jing Zhu, and Adwait Ratnaparkhi, IBM's Statistical Question Answering System, *TREC-9 Proceedings*, (2001).

(6) Dan Moldovan, Marius Pasca, and Mihai Surdeanu Sanda Harabagiu, Performance issues and error analysis in an open-domain question answering system, *ACM Transactions on Information Systems*, Vol. 21, No. 2, (2003), pp. 133–154.

(7) Yutaka Sasaki, Question answering as abduction: A feasibility study at ntcir qac1, *IEICE Transactions on Information and Systems*, Vol. E86–D, No. 9, (2003), pp. 1669–1676.

(8) Tetsuro Takahashi, Kozo Nawata, Kentaro Inui, and Yuji Matsumoto, Effect of structural matching and paraphrasing in question answering, *IEICE Transactions on Information and Systems*, Vol. E86–D, No. 9, (2003), pp. 1677–1685.

(9) TREC-8 committee, Trec-8 question answering track, (1999), http://www.re-search.att.com/~singhal/qa-track.html.

(10) NTCIR-3 QAC task committee, Question Answering Challenge (qac) home page, (2001), http://www.nlp.cs.ritsumei.ac.jp/qac/.

(11) Masaki Murata, Masao Utiyama, and Hitoshi Isahara, Question answering system using syntactic information, (1999), http://xxx.lanl.gov/abs/cs.CL/9911006.

(12) National Institute of Informatics, *Proceedings of the Third NTCIR Workshop (QAC)*, (2002).

(13) Toru Takaki and Yoshio Eriguchi, NTT DATA question-answering experiment at the NTCIR-3 QAC, *Proceedings of the Third NTCIR Workshop (QAC)*, (2002).

(14) 賀沢秀人, 加藤恒昭, 意味制約を用いた日本語質問応答システム, 情報処理学会自然言語処理研究会 2000-NL-140, (2000).

(15) Yuji Matsumoto, Akira Kitauchi, Tatsuo Yamashita, Yoshitaka Hirano, Hiroshi Matsuda, and Masayuki Asahara, Japanese morphological analysis system ChaSen version 2.0 manual 2nd edition, (1999).

(16) S. E. Robertson and S. Walker, Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval, *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, (1994).

(17) S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford, Okapi at trec-3, *TREC-3*, (1994).

(18) 村田真樹, 内元清貴, 小作浩美, 馬青, 内山将夫, 井佐原均, 位置情報と分野情報を用いた情報検索, 言語処理学会誌, Vol. 7, No. 2, (2000).

(19) Masaki Murata, Qing Ma, and Hitoshi Isahara, High performance information retrieval using many characteristics and many techniques, *Proceedings of the Third NTCIR Workshop (CLIR)*, (2002).