局所的類似情報に基づいた
テキストマイニングに関する研究

竹田隆治

博士 (情報学)

総合研究大学院大学
複合科学研究科
情報学専攻

平成 20 年度
(2009)

本論文は総合研究大学院大学複合科学研究科情報学専攻に
博士 (情報学) 授与の要件として提出した博士論文である.

審査委員会

高須淳宏 教授 (主査)	総合研究大学院大学/国立情情報学研究所
相原健郎 准教授	総合研究大学院大学/国立情情報学研究所
相澤彰子 客員教授	総合研究大学院大学/国立情情報学研究所
影浦峡 准教授	東京大学大学院
大山敬三 教授	総合研究大学院大学/国立情情報学研究所
市瀬龍太郎 准教授	総合研究大学院大学 /国立情情報学研究所

(主査以外はアルファベット順)

Text Mining Based on Locally Similar Information

Takaharu Takeda

DOCTOR OF
PHILOSOPHY

Department of Informatics
School of Multidisciplinary Sciences
The Graduate University for Advanced Studies (SOKENDAI)

March, 2009

A dissertation submitted to
the Department of Informatics,
School of Multidisciplinary Sciences,
The Graduate University for Advanced Studies (SOKENDAI)
in partial fulfillment of the requirements for
the degree of Doctor of Philosophy

Advisory Committee

Prof. Atsuhiro Takasu (Chair)	National Institute of Informatics/ The Graduate University for Advanced Studies
Associate Prof. Kenro Aihara	National Institute of Informatics/ The Graduate University for Advanced Studies
Prof. Akiko Aizawa	National Institute of Informatics/ The Graduate University for Advanced Studies
Associate Prof. Ryutaro Ichise	National Institute of Informatics/ The Graduate University for Advanced Studies
Associate Prof. Kyo Kageura	The University of Tokyo
Prof. Keizo Oyama	National Institute of Informatics/ The Graduate University for Advanced Studies

(Alphabet order of last name except chair)

論文要旨

現代は高度情報化社会といわれ，世界の情報の多くは電子化されている．こうした電子化された大量の文書が存在する一方，その中から必要とされる情報を効率良く見つけたいというユーザの情報取得要求も強くなっている．特に最新のニュースなどに関する記事は，非常にリアルタイム性が高く，情報取得の価値が大きいデータである．また，そういった事件事故のニュース，季節のごとのイベントや，話題の商品，サービスなど何かある事柄について，時間的に集中して web 上での意見の急増を見せることは非常に多い．web 上のコミュニティには エンドユーザの意見，要望，苦情，その他 企業側が想定しない潜在的 ニーズ，リスクなどが書かれている場合があり，こういった，CGM (Consumer Generated Media) など web 上での話題集積の重要性が増している．このような同一の話題について言及した文書中には，重複した情報が非常に多くあり，本質的にユニークな情報は少ない．しかも，その重複した情報とは，事前にはどのような情報が存在しているのか不明であり， 存在したとしてもどこにあるかがわからないことが多い．またそれ以前に，そもそも，どんな大きさで情報が存在しているのかが不明である．

既存研究の多くは，情報の単位として，文書レベル，文レベル，または何らかの文節など，事前にその単位を規定して，その枠組みの中，という狭い問題空間内での取り組みを想定している．本論文では，明示的にどこからどこまでに，どのような情報が存在しているのかが不明な問題を扱う．文書，文，などは情報の単位のひとつであるが，そのような明示的にわかる大きさの情報（意味）が全てではない．人間は，任意の部分（部分集合）に意味を見出すことがあるからであり，すべての部分に意味が存在し得ると考えなければならない．

ある話題についての情報を集めたいと思ったときに，web 上での情報収集は容易となり，要するコストも小さくなったが，ここで前述の問題が顕著に影響してくるようになった．例えば，ある検索エンジンを使ってある検索クエリーで検索をした場合，検索結果ページは，その検索ク

エリーに関する情報があることは期待できる．しかしながら，それらの中には，類似した，または完全に同じ情報が数多く存在する場合がある．同じ情報ばかりを何度も集めてくると，時間的，情報資源的に無駄が多くなるため，このような冗長性の削減の重要性も増してきているのである．本論文ではこの冗長性の削減が有効である具体的な応用問題と，実装したアプリケーション，そしてその評価にいたるまでを述べる．本論文を構成する具体的なアプリケーションを含んだテーマとして，複数文書要約，splog filter，この二つを取り上げた．

一つ目の応用問題として複数文書要約に取り組む．

あるニュース，イベントなどについての記事を世界各国の報道各社が一斉に書くことは頻繁にある．それらの記事の内容は，部分的に同じ情報を，異なる表現で，または完全に同じ表現で記述している部分が非常に多い．ある記事には部分的に新しい情報が存在しているかもしれないが，既知の情報も同時に含んでいることが多い．このような場合，その新しい情報の「部分」だけを列挙すればそのニュースについての情報は網羅的に知ることができると期待できる．この記事で，新しい情報以外の「部分」はこのユーザには必要ないのである．このような考えに基づき，同一のニュースクラスタに属する複数の文書から，自動的に新たな単一文書を生成する手法とそのシステムを実装した．

二つ目の応用問題として Splog filter に取り組む．

任意の一致した情報を持つ文書として，splog というブログエントリがある．近年，web 空間上の資源を圧迫し，検索エンジンの検索結果にも多大な悪影響を及ぼしている．Splog を検出する方法として様々な手法が提案されているが，本論文では，その中の任意の一致に注目した検出法を提案する．Splog は様々な方法で生成されるが，splog はその生成方法から，コンテンツに同一の文字列が存在する場合が多くある．後述する content snatch や search results が，同一の元コンテンツコピーしてきた場合には，当然，生成される splog コンテンツも同一のものを含む． Template decorator(後述)が同じ template を使えば，同然，生成される splog コンテンツも同一のものを含む．この場合，content snatch ,search results では偶然にもコンテンツが一致することになるのであるが，template decorator の場合は必然的にコンテンツが一致することになる．このような傾向から，このコンテンツの一致から splog の検出ができると思われる．この一

致は，どのブログエントリの，どの部分に，どのような大きさで存在するかはわからない．局所的類似性はここに存在し，本論文の主題はここに存在する．Splog の場合は類似ではなく完全一致であるため，これは局所的類似性の特殊なケースであり，一例である．

Abstract

Various kinds of information are currently accessible through the Internet. Since the Internet contains a wide variety and a large amount of information, an efficient and effective information acquisition mechanism is strongly required. News articles are one of the most useful information sources because they contain up-to-date information about various kinds of topics, such as serious accidents, seasonal events, popular merchandise, and services. Consumer Generated Media (CGM) data such as blog is another example of important information, because it contains valuable information concerning a customer's reputation and becomes an important information source for detecting customers' needs and analyzing the effects of various product promotions. Although the Internet consists of a huge amount of information, it also contains a lot of duplicate information. The duplication is harmful when handling the information. However, we could not know what and where there are beforehand.

For example, suppose we search the Web pages relevant to a given topic. Then, the search result may contain pages describing the same thing. The duplication is a nuisance when trying to comprehend the search results. It is also problematic from the aspect of computational and network resources. By removing the duplication, we can make the Internet a more easy-to-handle information source.

Existing problem is that the duplication appears in configured levels, e.g., words, phrases, sentences, and paragraphs. This thesis describes a new method for removing duplicated information. For this goal, we first develop an algorithm to enumerate the locally similar information that is defined as the substrings of any length that appear frequently in multiple documents. Since there is a vast amount of substrings in documents, in addition, the duplicated information may be described using different expressions. We focus on the efficiency of the algorithm. We applied it

to two kinds of applications to evaluate the effectiveness of the developed method.

The first application is a multi-document summarization of news articles. News articles are delivered from multiple news companies, and we can read most of them on-line. So, news articles describing the same event issued by the different companies exist. Furthermore, news articles are frequently updated and parts of news articles are duplicated even if they are issued by the difference company. In this case, Summary article can be created if the new information "part" are enumerated. Our news summarization system detects the locally similar information from news articles, makes clusters of them describing the same event, and generates a summary of each cluster.

The second application is a splog filter. Splog is a blog that is generated automatically for commercial purposes. They are harmful for CGM content retrieval and analysis. Japanese splogs are often generated by combining words and copied phrases appearing in various documents. As a result, Japanese splogs contain copied words, phrases, and sentences . These copied strings are regarded as locally similar information.

This thesis exploits this feature of Japanese splogs and proposes a splog filtering system using the proposed method of enumerating locally similar information and shows that it is effective for filtering splogs. This is a special case of local similarity because Splog do not have similarity but the exact match .

目 次

表 目 次

図 目 次

第1章　序論

1.1　問題設定と背景

web 上の電子的情報にはほとんど同じ，あるいは完全に同じ情報が無数に存在するが，近年の情報の電子化と，インターネットなどの電気通信系パーソナルメディアの伸びなどが情報流通コストを大幅に削減したことにより，同質，同等の情報が大量に存在するという問題が表層化してきた．このような問題は情報検索において顕著に影響してくる．例えばユーザが検索エンジンなどを通して検索結果のコンテンツを閲覧する際，それらの間に重複情報が存在すると，既知の同じ情報を何度も見ることになり，情報取得に要する時間的効率が悪くなる．情報検索の研究においてもこの点は問題とはしている．しかしながら情報検索の側で問題としている情報の単位は，例えば web ページなど，明確にその大きさが区切られたコンテンツとしての情報量であり，取り扱い，ユーザに対して提示されるべき「情報」の最小単位は，web ページなど，単一文書のレベルである．本研究で問題とする対象は，例えば web ページなど文書内での部分情報である．

ある web ページ全体の情報量が小さいからといって，その web ページ内全ての情報が無駄で役に立たないということには必ずしもならず，また逆に，情報量が大きい web ページ内の全ての情報に新規性があり，すべてが重要であるということにはならない．部分情報とは，例えば,「…なので，”X が Y である”」「”X が Y である”が，しかし…」「”X が Y である”と，言った」「”X が Y である”ことは知られ…」という部分文集合があるとする．ここには，”X が Y である”(X is Y) という表現が共通して存在している．ここでは，その表現の目的などとは関係なく，そのような表現が指し示す共通の意味（記号）が存在していると思われる．もちろん，”X が Y である”という短い表現であれば，偶然，必然を問わず頻出すると思われるが，もっと長い表現でも同一の意味に解釈できる表現は数多い．また，完全に同一の文字列でなくとも，別の言い回しで同

一の意味を指し示す表現などを含めれば，類似する部分情報（部分記号）というものは確かに存在していると言える．ある任意の表現が，コーパス中に一度しか存在しないのであれば，そこには意味は存在しないと考えてもよい場合はあるが，任意の大きさの類似した表現が，必然であれ偶然であれ複数回出現するのであれば，それらは分割（抽出）可能な何らかの「情報（記号）」を指し示していると考えられる．その「情報」が，文脈によって意味と目的とが変化することとは独立に，そのような共通の「情報（記号）」がそこには存在している．

本来，人間の認識・知覚レベルにおいて，情報量を計算すべき情報の単位は，明示的に区切られているものではない．人間にとってなんらかの意味のある「情報」は，すべての分割可能な「部分」に存在する可能性があると考えるべきである．

1.2 本研究の目的と意義

Web 上の問題に限らず，物理的世界，人間社会において，ある話題についての情報を集める際，それらの中に同じ情報が何度も現れることは多い．ある個体には部分的に他と異なる情報が存在しているかもしれないが，共通の情報も同時に含まれていることが多い．このような場合，その共通する情報の「部分」だけを列挙すれば，その話題についての情報は網羅的に知ることができると期待できる．このような部分類似情報を調べることで，この集合の中には，どのような情報の塊，集合，クラスタ，が存在しているのかを，事前知識なく，自己組織的に検出できる．逆に，そのような共通情報を冗長とみなし削除することで，過不足ない情報の集積も可能となる．これによって，既知の，同じ情報を何度も見ることなく，効率的な情報収集を可能にすることができる．

本研究は，このような局所的類似性を効果的，効率的に検出する手法を提案し，具体的な応用問題へ適応し，その有効性を示すことを目的とする．

第2章　関連研究

2.1　概要

本研究の関連研究として，次のような研究分野が挙げられる．

1. 文書類似性尺度
2. 文書要約
3. アラインメント
4. スパムフィルタ
5. 実際のサービス，商品

スパムフィルタ，文書要約の関連性は自明である．それぞれ，4 章，5 章で取り組む．

アラインメントは，記号配列中に存在するかもしれない任意の類似性を自己組織的に検出する概念として，本研究に対して全般的に関連性があり，文書類似性尺度も，本研究では個別の具体的な各式に関連がある．

実社会への影響，応用，実用化を踏まえて，実際にどのようなことを狙ったテキストマイニング技術製品，サービスが流通しているのかも関連として加えた．

2.2　文書の類似性

情報検索，文書分類などの問題で古くから用いられている基本的で有名な手法として cosine 距離がある．文書をベクトルで表し，その内積を距離，類似度として用いる方法である．

単語の重み付けの古典的な方法に TF-IDF があり，文書中の各単語の TF-IDF 値計算し，値でソートすると，その文書に特徴的な単語リスト

を得ることができる．TF-IDF は，単なるヒューリスティックスだと考えられていたが，最近言語モデルに基づく情報検索手法がさかんに研究されるようになり，TF × IDF の解釈が明らかになってきた．言語モデルに基づく手法は，ヒューリスティックスによる手法と同性能にもかかわらず，文書のランキングに理論的で合理的な説明を与えることができる．[1]

2.3 文字列の類似性

異なりを許容した類似一致にもいくつか考えがあり，[37, 40, 2, 32] 文字列の並び順を考慮した類似度としては，編集距離 (Levenshtein distance) がある．二つの文字列（記号列）がどの程度異なっているか，その文字列の編集に要するコストで表現する値である．コストの与え方によって様々な応用問題への適応が可能だが，近年ではバイオインフォマティクスの分野で活用されてきている．本稿では，文字列の重みを動的に決定するために，主に類似度の評価として，重み付き編集距離 [15] を用いる．

2.4 アラインメント

2.4.1 相同性解析

アラインメントは，バイオインフォマティクスにおいて遺伝子の解析に用いられる手法，概念である．

遺伝子において， 相同性とは， 共通の祖先遺伝子から由来していることを意味している．遺伝子の一部は子孫に必ず残っていくため，遺伝子 A と遺伝子 B が共通の先祖遺伝子から由来している場合，これを示す類似領域が A と B の遺伝子配列中に散在していることが多い．

類似した記号列の検出ができれば遺伝子の相同性を示せるが，この時どのような記号列がどこに存在するか，またはそもそも存在しないかは事前にはわからない．アラインメント [16, 7, 39, 6, 35] は，このように複数の記号列中の類似している領域を効率的，効果的に検出する手法として用いられている．遺伝子の塩基配列そのものや，アミノ酸配列を調べることが一般的である．

[1] http://chasen.org/~taku/blog/archives/2005/11/_tfidf_1.html/

ローカル (局所) およびグローバル (大域) アラインメント

グローバルアラインメントは，二つの配列間で，始点から終点までの類似性を実数値で計算する．類似性は，順序付き記号列間で定義できる距離であればどのような定義でもよいが，ほとんどの場合で編集距離のバリエーションが用いられている．

スコアについては，遺伝子集合の特徴にあわせたモデルが導入されている．変異が，配列中のすべての位置で互いに依存せず独立に生じることを前提にしている．連続するギャップ（挿入・欠損）に対するペナルティスコアの計算方法として，リニアギャップペナルティ と アフィンギャップペナルティ がよく用いられるリニアギャップペナルティでは n 個のギャップがある場合，n に線形比例したペナルティを課すが，アフィンギャップペナルティでは， n の大きさよりも， そのギャップが存在すること自体に対するペナルティを課すという考えである．塩基配列は 1 個ずつの塩基の変化（個体変異による）よりも，連続した長い領域の変化（交叉による）の方が起こりやすいためである．そのため，変化の長さ自体はそれほど重要ではないと考える．

グローバルアラインメントは配列全体の類似度を計算するのであるが，ローカルアラインメントは，配列中の部分的に類似した領域を特定し (図 2.1)，配列間でその対応づけを行う．

類似していることが予想される領域が特定できていれば，その領域に対してグローバルアラインメントを行えばよいのであるが，先に述べたように,（相同性があるかもしれない）配列の組のどこが類似しているのか，そもそも，本当に相同性があるのかどうかは解析前にはわからない．

例えば， 複数のタンパク質が共通の領域を持っていたり， 広い範囲の塩基配列の断片を比較する場合などに意味がある．例えば， 2 本の配列がその全体にわたって共通の祖先から進化したものではあるが，それらが非常に離れてしまった場合の配列の比較に適用できる．

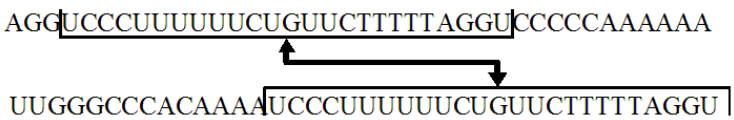

図 2.1: ローカルアラインメント

ペアワイズおよびマルチプル (複数配列) アラインメント

アラインメントは基本的に 2 配列の間で類似部分の対応づけを行う．上記のグローバルアラインメント，ローカルアラインメントは，どちらも基本的にはペアワイズアラインメントであり，２配列を対象とすることを前提にしている．マルチプルアラインメント (図 2.2) はペアワイズアラインメントの拡張であり，3 配列以上を扱うもので，進化的に保存された配列の同定などに用いられる．マルチプルアラインメントには，累進法，反復改善法などの手法がある．

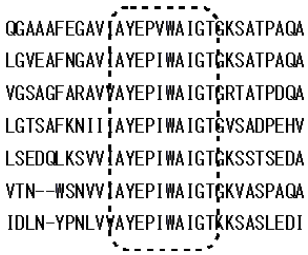

図 2.2: マルチプルアラインメント

2.4.2 CLustalW

ClustalW [16] は 反復改善法によるローカル・マルチプル・アラインメントアルゴリズムである．ClustalW では，全ての配列の組み合わせに対してペアワイズアラインメントを行い，配列一致度の行列を作成する．この一致度に基づいて樹形図 (図 2.3) を作成し，距離尺度を用いて階層型クラスタリングを行う．この際，近隣結合法，または非加重結合法が用いられる．

樹形図に従って，最も一致度の高いペアから開始し，樹形図に沿って 1 つずつ配列を追加しながら整列させていくことで効率的に多重整列を得る．

反復改善法では， 記号列をいくつかのグループに分割し，そのグループ内でのアラインメント結果同士を統合していくことによって全体のアラインメントを行う手法である．アライメント結果が収束するまで，グループの再分割，再統合を繰り返す．

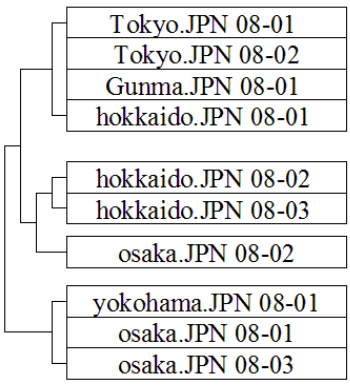

図 2.3: 系統樹例

自然言語処理におけるアラインメント

本稿で提案するアラインメントは，自然言語処理 [44] [55] で一般的に使われているアラインメントとは若干意味が異なる．

機械翻訳で使われるアラインメントは，多言語間で同じ意味を持つ単語，語句を対応させることを意味する．本稿の場合は，基本的に同一言語内での類似表現対応を抽出し，語句レベルよりももっと長い領域の対応づけを行う．繰り返しになるが，あるパターンに対する類似領域をコーパス中から検出する [37, 40, 32] わけではない．どことどこが対応づけられるのかを自己組織的に検出するのである．

もう一つ，木構造のアラインメント [38] という概念もある．編集距離の考えの拡張として，木の編集距離と木のアラインメントを導入し，HTML, XML 形式などの構造データや，文の構文構造のアラインメントを行い，頻出表現の抽出などを行うものである．

本稿の場合は，構造として単純な順序つき記号列を対象とし，構文解析を用いたりしない．考えとしては丸川ら [57] の研究が最も近いのであるが，加えて，マルチプルアラインメント，つまり三つ以上の文書を同時にアラインメントをするという点で異なる．本稿では，これらのアライメント技術を自然言語処理に導入するが，語句，単語の出現頻度という自然言語特有の特徴を用いることでアラインメントを効果的に行うところに特徴がある．

2.5 自動要約

2.5.1 要約生成

文書要約手法は大別して，意味解析による要約生成 [3, 36] と，文抽出による要約生成 [5, 29, 9, 23, 25, 52] とに分類できる．複数文書要約のためには重要な情報を選択しなければならないが，複数の文書で同じ情報が記述されていることがあるため [58]，重要な情報を取得するだけでは不十分で，冗長性除去が重要な課題になる．

文生成

Barzilay ら [3] は，複数の文書にわたる同様の意味要素を統合することで「簡潔」な要約を作成する方法を提案している．まずはじめに事柄の結合方法を決定し，それに基づいてこのクラスタにおける「主題」をそれぞれ，一つの文に統合していく．Columbia NewsBlaster [22] [2]は，文献 [3] の拡張バージョンを使って要約している．これは，入力文書中の類似文を，機械学習と統計的手法 [10] によって特定する．

吉岡ら [36] はイベントモデリングによる複数文書要約を提案している．イベントの要素となる語句の選択によって要約を生成するという意味解析による要約生成である．ここでも，各要素の重複を調べることで冗長性をチェックするという手法をとっている．

文選択

Centroid-based summarization (CBS) [28, 29] ではニュースクラスタの中心となる文を抽出するため，その重心 (centroid) という概念を導入する．重心とは，この文書クラスタにとって統計的に重要と考えられる単語ベクトルであり，この重心に近い文を選択することで要約生成を行う．この際に，文の包括性という考えを導入し，他の文に包括される文を削除することで冗長性を削減する．

Carbonell らは，MMR(Maximal Marginal Relevance)[5] を用いた要約法を提案している．MMR は情報検索において広く用いられている手法である．文書を，クエリーとの関連度と内容の新規性の組み合わせでラン

[2]http://newsblaster.cs.columbia.edu/

キングし，すでに選択された文に似た文にマイナスのスコアを与えることで，新情報を含む文を抽出し，要約を生成するものである．

森ら [23] は質問文を知識として使用して要約文を生成する手法を提案し冗長性削減には MMR [46] が用いられている．

岡崎ら [25] は information fragment という概念による複数文書要約を提案している． 構文解析によって文を information fragment という要素に分割し，この要素を多くカバーする文を要約要素に選択することで要約生成を行う．この際，各文ごとに information fragment の重複をチェックすることで冗長性を調べるというアプローチである．

2.6 スパムフィルタ

2.6.1 Splog とは

ブログは個人の日記としての情報が記されており，商品やサービスに関する消費者の体験談や感想，生の声を収集する上で有益なメディア [3] として注目を集めており [45]，ブログを対象とした研究も盛んである [48, 54, 49, 8, 43, 41] [4] ．

しかし一方でブログコンテンツの作成は容易で，商用利用を目的とした splog [5] と呼ばれるスパムコンテンツが増加している．splog は情報検索品質を低下させ，web アーカイブ資源を浪費させるという問題を引き起こしている．splog とは，スパム (spam) とブログ (blog) を合成した造語で，インターネットにおいて，リンク誘導や広告収入などの目的で生成される．本稿では，このような自動的に生成される splog のフィルタリング手法について論じる．

splog は他のコンテンツから語やフレーズをコピーし，それらをつなぎ合わせて生成されることが多く，splog の内容はその時々のニュースや流行によって変わるため，スパムメールフィルタで使われるベイズフィルタのような一般的な統計的手法で対処することが難しい．また，検索エンジンで上位の検索結果コンテンツ (web ページ) からコピーしてくるなどの方法によって，splog 自身の検索順位を上げようとしている．複数のコンテンツを部分的にコピーしてつなぎ合わせることで検索エンジンに「類似したページ」と認識されないようにしている．

[3] http://en.wikipedia.org/wiki/Blogosphere/
[4] http://blogreport.labs.goo.ne.jp/
[5] http://en.wikipedia.org/wiki/Splog/

加えて，splogが他のwebページへのリンクを大量に生成するため，ページランク [4, 26] の変動が生じ，検索品質に著しい悪影響を及ぼしている．

splogのフィルタリング手法は，リンク解析による手法とコンテンツ解析に手法に大別できる．

2.6.2 コンテンツ解析によるフィルタリング

Kolariら [13, 12, 14] は，英語のブログを対象にsplogフィルタリングの先駆的研究を行った．特定種類の単語の割合を特徴量ベクトルとして表現し，SVM(Support Vector Machine) などの機械学習の手法を用いて判別を行うコンテンツ解析手法を提案している．

日本語のブログを対象とした研究も始められているが，英語のsplogフィルタリングのように文書を特徴ベクトルとして表現した分類問題とは異なるアプローチがとられている．成澤ら [51, 24] はコンテンツ中に頻出する文字列に着目した．ブログエントリ中の部分文字列の出現頻度と異なり数には一般的にジップの法則が成り立つが，splog中には，検索クエリーとしてよく使われる単語，最近話題の言葉，注目を集める言葉など，ジップの法則からはずれる語やフレーズが多く含まれる．そこで，これらのsplogに特有な語やフレーズを抽出する方法を提案した．

2.6.3 リンク解析によるフィルタリング

リンク解析を用いた手法として，Linら [18] は，いわゆるリンクファームと呼ばれる大量の不自然な被リンク群に注目したsplog（群）の特定手法を提案した．石田 [42] も基本的に同じアイデアでリンク構造からのsplog検出を試みている．コンテンツ解析とリンク解析を併用した手法として，Linら [19] は，ブログエントリの自己相関からsplogを検出する手法を提案した．splogは，何回も繰り返し同じコンテンツを作り出すことが多いため，リンク先，タグ，タイトルなどの相関行列を構成し，これを特徴量としてsplog特定を行っている．

2.6.4 その他のフィルタリング手法

splog filtering手法は，基本的には，検索エンジンの検索上位に位置することを狙った不自然なコンテンツに顕著な傾向に着目することでフィルタ

リングを行おうとする．splog filteringは，その問題の新規性と特殊性から，さまざまなアプローチが提案されており，URLの文字列に着目したsplogフィルタリング法も提案されている．例えば， www.dietthatworks.comというドメイン名は，diet that worksの3つの単語を結合して作られている．Slabetti [30]は，ドメイン名を構成する各単語のsplog率を定義し，ベイジアンフィルタを用いて分類する方法を提案した．筆者らの調査では，日本語のsplogにも，“youtubemovie”， “affiliateinfo” といった単語の組合せによるドメイン名を使ったsplogが見受けられたが，“skdsjdljdks” や“g3n0x9h2ja9y1” といったランダムな文字列を用いたドメイン名の方が多かった．

本論文で提案するフィルタリング手法は，コンテンツのコピー検出に基づいてフィルタリングを行うものであり，着眼点は成澤ら [51, 24] の手法に近い．成澤らの研究はスパムテンプレートの発見に焦点があてられているのに対して，本研究ではコピー文字列の検知に焦点が当てられている．そのため，スパムテンプレートが特定できなくとも，コピー文字列の検出によりsplogを検出することができる．

2.7 システム

2.7.1 マイニングシステム

蓄積文書向け

古くからテキストデータに対する様々なマイニングシステム [60, 59, 31] が開発されてきた． 例えばコールセンターへの顧客の問い合わせ，アンケート結果などの自由記述による膨大なテキストデータから，何らかの単語（単語の集合）の量や，変化量などを分析し，その結果をマーケティング調査やブランドイメージ調査，解約防止分析，品質管理，顧客満足度調査，商品開発など，サービスの改善などに役立てるためのシステムである．

こういったマイニングシステムでは形態素解析，構文解析 (係り受け解析) など，自然言語処理に必要な基本機能などに加え，共起頻度や出現頻度の偏りなどからの関連語分析なども可能なことが一般的である．時系列解析，クロス集計，属性別分析，などといった統計的分析機能はほぼすべて網羅しており，製品によってはより高度な，特徴的な分析手法を採用していることもある．

CGM メディア向け

近年は CGM メディアを利用したマーケティングリサーチが重要視されてきており，kizasi.jp [6]， goo 評判解析 [7]， 電通バズリサーチ [8] など web マーケティングツール も数々開発されている．蓄積文書向けと異なる点は，対象となるテキストデータが web 上の電子掲示板やブログなどといった不特定多数の情報源ということである．

もうひとつ異なる点は，蓄積文書向けツールの場合は，自社で分析対象となるテキストデータを持っている必要があるが，CGM メディア向けツールの場合はクライアントがそのデータ収集を行う必要はない．サービス提供企業が web 上のテキストデータを収集し，クライアントは Web 経由で分析ツールにアクセスする方式が一般的である．

基本的な解析機能は上記の蓄積文書向けとほぼ同様であるが，テキストデータの情報源別（電子掲示板/ブログ）の選択機能や，スパムフィルターを搭載していることが一般的である．

2.7.2 News aggregator

最新情報を自動的に収集，提示する代表的なシステムとして Google News [9] をあげることができる．Google News は，複数のニュースソースから記事を収集し，類似した内容の記事をグループにクラスタリングしている．同一の話題について複数の情報源からの記事を関連付けるという基本機能に加え，サイトレイアウトの編集や，ユーザ毎の最適化などといったインタフェースレベルの技術的工夫が非常に数多くある．

一方，NewsBlaster [22] は，文書の差異に注目した要約手法の MMR を用いた複数文書要約システムである．記事のクラスタリング機能は Google News と同じであるが，人間の編集作業を介することなく自動的に要約文を生成するという点が特徴になっている．

Web サイトの構築，運用コストの低下，プロトコルの普及と一般化により，小規模な工数でも自動ニュースシステムの開発が行えるようになっている．企業，組織レベルではなく，個人的にも自動ニュースサイト [10] を開発し，運営している者は数多い．

[6] kizasi.jp http://kizasi.jp/
[7] goo 評判解析 http://buzz.search.goo.ne.jp/
[8] 電通バズリサーチ https://www.dbuzz.jp/
[9] http://news.google.com/
[10] http://news.ceek.jp/

第3章　局所的類似表現の抽出

3.1　局所的類似性

局所的とは，任意の分割可能な情報（個体）中の，事前に明示的に分割可能性が示されてはいない任意の「部分」のことである．類似性とは，任意の可能な解釈による類似性である．類似しているかどうかの解釈，類似性の定義はどのようなものであっても構わない．

このような類似性は，様々な情報源の背景，性質，独立性，などにはほとんど無関係に生じる．意図するか意図しないか，偶然か必然かによらず，その情報の扱いの重要性，方向性,（長期的／短期的）目的，などにも依ることなく，である．

このような現象は物理的実世界でも普遍的に生じるが，情報技術の発達によって，web 上の言語空間においてこのような問題が顕在化し，要望，ニーズも同時に大きくなっているのである．

情報検索を例とする．web ページなどの単位を対象として扱っているが，ユーザは，その web ページのすべての情報を知りたいわけではなく，その web ページのすべての情報がユーザにとって役に立つわけではない，ユーザは知りたいことだけを知りたいのであり，知りたい情報以外はユーザには必要ない．検索結果として得られた web ページの一部にはユーザが知りたい情報があるかもしれないが，他の部分はユーザにとって有用ではない情報かもしれない．

局所的類似性とは，複数の情報源のそれぞれの一部分（局所）同士が類似している現象を指す．例えば，検索結果 web ページ中で，ユーザの要求に応える ”部分 ”それぞれには，局所的類似性があると思われる．

3.2　局所的類似性の定義

同一の事柄について記述した文書では，ある程度以上の長さの一致する語句，分節が高い確率で存在し，並び順もほとんど同じく記述されてい

る場合が多い，例えばニュース記事などでは完全に同一の文が存在することは多い．bag of word 形式のような語句の集合ではなく，語句の並び順を考慮した類似度が，このような文書中の部分情報の類似度を計算する上では有効である．提案手法では，閾値 $threshold$ と， 一致長 $length$ という二つのパラメータによってこの類似度を制御する．要素 S_i, S_j は，順序付き記号列（文字列，単語列）であり，C はこの記号列の集合である．

本研究中で扱う局所的類似性は，以下のように定義する．

$$(\forall i)(\exists j) func(S_i, S_j, threshold) \geq length \tag{3.1}$$

ここで， S_i, S_j の先頭から l 文字分の部分文字列 を S_{il}, S_{jl} とする $distance(S_{il}, S_{jl})$ を， S_{il}, S_{jl} の距離関数とする．

$$func(S_i, S_j, threshold) = \max_{l} \quad (distance(S_{il}, S_{jl}) < threshold) \tag{3.2}$$

$func(S_i, S_j, threshold)$ は， S_{il} と S_{jl} の距離が，$threshold$ を超えない最大の l である．これは， S_i と S_j の長さ l のプリフィックスが十分に類似しているということである．

すべての $S_i \in C$ に対して，$func(S_i, S_j, threshold) \geq length$ を満たす $S_j \in C$ が一つ以上存在する．このような部分文字列集合 C は局所的類似性があるとする．

これは，単純な単連結法 (single link) によるクラスタに近いが， 任意の要素は複数のクラスタに属してもよい．現実的にはこのようなクラスタをすべて列挙するには非常に大きな計算を要するため，高速で（有用な）近似解を求める手法を提案することが，本稿の要点である．

3.3 既存の問題設定との差異

類似している要素の集合を自己組織的に構成するという点ではクラスタリングは近い問題であるが，本研究と既存のクラスタリング問題との差は，クラスタリングするべき要素が事前に規定されていないという点にある．前述のとおりバイオインフォマティクスにおけるローカル・マルチプル・アラインメントが，本研究により近い問題である．既存のローカル・マルチプル・アラインメントとの大きな差は，階層的なアラインメントを行ったり，逐次的な修正をしたりせず，一度の実行で複数配列のアラインメントを完了させるという点である．

通常のクラスタリング問題の場合は，集合を構成すべき要素が事前に列挙されているため，これらの最適な集合を構成すればよい．要素の数を n とすれば可能な組み合わせの総数は 2^n である．

これだけでも膨大な数ではあるが，本研究が対象とする問題空間はさらに巨大である．通常のクラスタリングで扱う場合には全ての部分文字列，つまり文書数を D，各文書の最大長を L とした場合，そのすべての始点と終点の組み合わせ $O(DL^2)$ の全要素を用意する必要があり，コーパスの規模が大きいと記憶容量が足りなくなる．そして，さらにそれらの組み合わせ冪集合の中から最適な集合を選択しなければならない．可能な集合の大きさは $O(2^{DL^2})$ である．

本研究ではこのように事前に要素は用意しないが，事実上，すべての要素が事前に存在しているように仮定し，自己組織的に部分文字列の始点と終点の決定，つまり要素の選択も同時に行う．本研究は，文書中の任意の「部分」に注目する．

3.4 局所的類似性検出アルゴリズム

本研究での手法の各プロセスにおいて，例えば文字列配列構築の順番や，類似度定義などで目的に合わせて変化を作れる．この定義，アルゴリズムはその基本形である．

前節で述べたとおり局所的類似性の定義を満たすクラスタをすべて列挙することは計算量的に困難であるため，高速で近似解を求める手法が必要である．

3.4.1 文字列配列構築

まず以下に示すように全ての可能な部分記号列集合 S を取得する．

ある記号列集合が $\{D_1, D_2, \cdots, D_n\}$ で構成されているとし，D_i は 記号列（文）$\{W_{i1}, W_{i2}, \cdots, W_{im_i}\}$ で構成され，記号列 W_{ij} は記号 $w_{ij1}w_{ij2}\cdots w_{ijl_{ij}}$ で構成されるとする．S を記号列集合中の任意の文の部分記号列とする．

$$S \equiv \bigcup_{i,j} \bigcup_{1 \le k \le l \le |W_{ij}|} \{w_{ijk} \cdots w_{ijl}\} \tag{3.3}$$

ここで，$|W_{ij}|$ は，記号列 W_{ij} に含まれる記号列長を表す．

1. $T \leftarrow \{\}, Cand \leftarrow \{\}$
2. for $i = 1$ to $|S|$
3. $\quad W \leftarrow S(i)$
4. $\quad C_m \leftarrow arg\ max_{C \in Cand}\, func(W, C, \tau)$
5. $\quad$ if $func(W, C_m, \tau) > e$ then
   $\quad\quad C_m \leftarrow C_m \cup \{W\}$
   $\quad$ else
   $\quad\quad Cand \leftarrow Cand \cup \{W\}$
6. $\quad$ if $|Cand| > \lambda$ then
   $\quad\quad C_f \leftarrow arg\ min_{C \in Cand}\ func(W, C, \tau)$
   $\quad\quad Cand \leftarrow Cand - C_f$
   $\quad\quad$ if $|C_f| > 1$ then
   $\quad\quad\quad T \leftarrow T \cup \{C_f\}$
7. $T \leftarrow T \cup \{Cand\}$
8. T を返す

図 3.1: クラスタリングアルゴリズム

この S 中の記号列の索引を作成する．集合 S に含まれる任意の部分記号列 W に対して，$\leq$ を 導入し反射律，推移律，対称律， 完全律を満たす全順序集合 $(S, \leq)$ を構成する．

S 中の任意の要素には番号が付けられ，大小関係がある．記号の単位，$\leq$ の定義は問題によって変えることができる．

3.4.2 クラスタリング

提案手法の要点は，同時に，自己組織的に 一対多の類似度計算を行うことであり，記号列配列を一度の走査でマルチプルアラインメントを行うというところである．

アルゴリズムへの入力として索引リスト S を与える．出力は単語列クラスタ集合 T である．また，S 中の i 番目の単語列 W を $S(i)$ とする．

クラスタリングアルゴリズムを 図 5.4 に示す．これは，記号列集合 S 中の記号を，先頭から順に

e はこのクラスタの最低類似一致長であり，3.2 章における l に相当する．τ は，距離 $threshold$ に相当する．λ はクラスタ候補数の上限である．

このようにして T を得ることが提案手法の要点であり，得られた T をどのように用いるかは応用問題によって変化する．また，それぞれの問題別に $distance()$ の定義を変えることによって対応が可能である．

3.5 個別の適応問題

本稿が扱う応用問題である複数文書要約と splog filtering に共通するのはこのような「部分」の類似性検出である．

本稿の手法を適応する問題としてこれらを取り上げた理由であるが，まず現代社会での問題の大規模化と，その解決の要望である．高速の情報化，大量の情報化に伴って問題が顕在化し，同時に社会的ニーズも強くなっているということが理由である．2 番目にはデータの収集の容易さ，収集に要するコストの低さである．比較的容易に保存することができるからである．

そして 3 番目に問題の適性であり，本研究の提案手法が有効に機能することが期待できるコーパスであるということが理由である．大規模なテキストデータ中には，類似または同一の情報が頻出する．ここでの「情報」とは，すべてのレベルにおける情報である．形態素，文節，部分文，文，段落，文章などといった言語的情報や，主張，予想，希望，感想，など，意味に関する情報も含める．特に，何らかの物事に関する新聞記事や，ブログの文中にはこれが極めて顕著に現れる．ニュース記事などは同じニュースを，同じ ような文面で，同じ ような内容を横並びで伝え，他紙との差異が明示的には示されない．ブログでもこれはほぼ同じであり，加えて，同一時期に集中的に類似する情報が記述されるという現象が継続的に起こり続けている．

文書要約の場合，同一の話題について言及された文書中に類似する表現が現れることは当然である．図 3.2，図 3.3 のように文書全体が類似した意味を伝える場合は多いが，実際にはその中の部分にこそ類似性がある．

どのような扱いをするか，どのような言及をするか，その話題の，どの部分を取り上げるか，どのように取り上げるか，などによって様々な違いが生まれるが，同時に，偶然と必然とに関係なく共通の「部分」が生じる．図 3.4 ， 図 3.5 の例では，類似した情報を色分けで示した．

それらは完全に同一であったり，言い回しが異なったり，あるいは完全に別の表現で同一の意味をつたえているかもしれないが，そこには局所的類似性が現れる可能性が高い．

政府・与党は13日、定額給付金など追加経済対策の裏付けとなる2008年度第2次補正予算案の今国会提出を見送る方向で調整に入った。今月30日までの今国会会期は延長しない方針だ。

政府、与党は13日、総額2兆円の定額給付金など追加経済対策を実施するための第2次補正予算案と関連法案の提出について、来年1月の通常国会に先送りする方向で最終調整に入った。

政府・与党は、定額給付金など追加景気対策を盛り込んだ2008年度第2次補正予算案の今国会への提出は見送り、来年1月召集の通常国会で処理する方向で調整に入った。

政府・与党は、定額給付金など追加景気対策を盛り込んだ2008年度第2次補正予算案の今国会への提出は見送り、来年1月召集の通常国会で処理する方向で調整に入った。

図 3.2: 類似表現 例 1

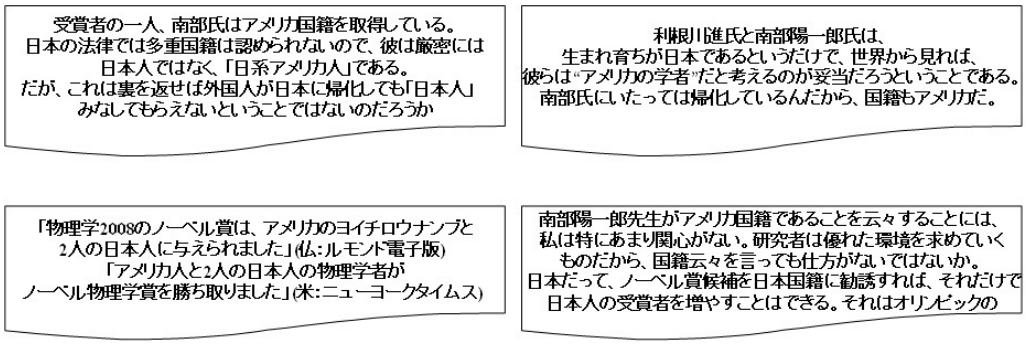

図 3.3: 類似表現 例 2

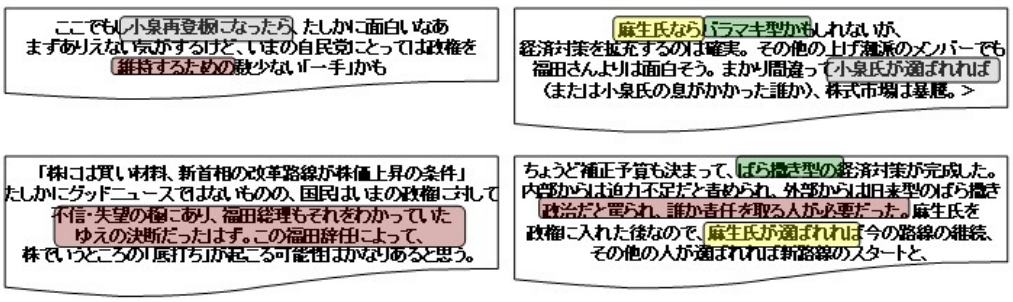

図 3.4: 局所的類似性 例 1

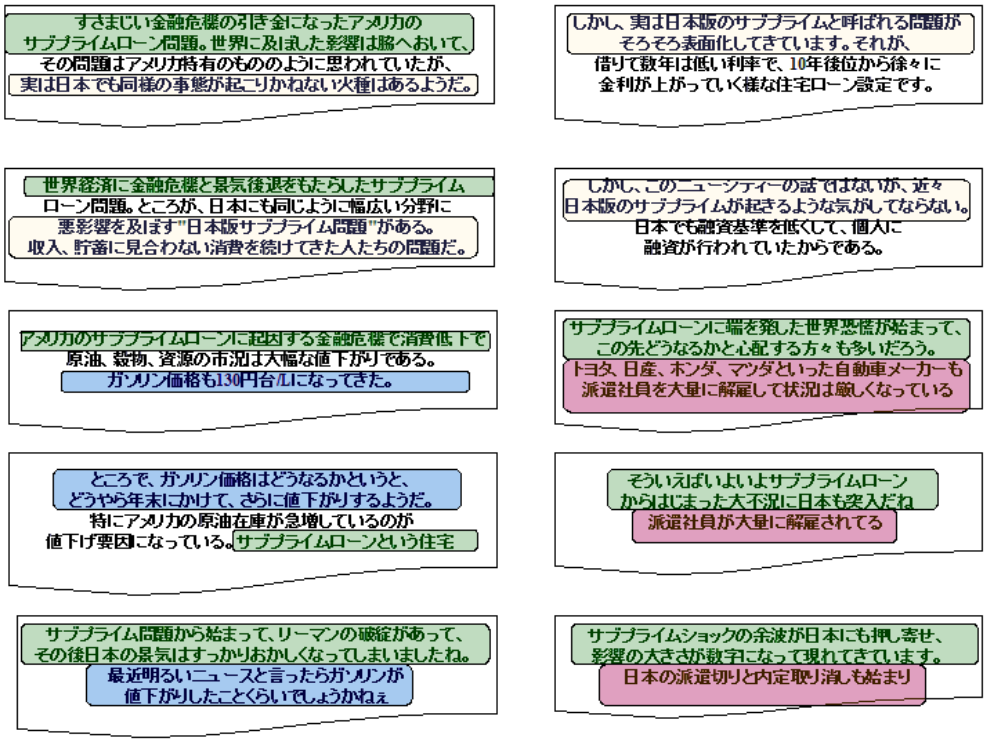

図 3.5: 局所的類似性 例 2

複数文書要約の場合は T を要約要素として用いる．$C \in T$ という類似部分文字列が存在していることが得られたので，C を 2 回以上記述しないように要約文を生成するのである．

splog は，コピーコンテンツを用いて生成されるため，複数文書に表れる文字列を見付けることによって splog を検出できると考えられるが，どのブログエントリの，どの部分に，どのような大きさで，どのような一致が存在するかはわからない．splog の場合は文字列がコピーされるため類似ではなく完全に一致する部分文字列を検出する必要があるが，これは局所的類似性の特殊なケースとみなすことができる．splog filter の場合は T をコピー領域とみなし，これが占める割合が大きいほど splog の可能性が高いと考える．両問題ともに共通した特徴は，どこにどれほどの大きさのどのような $C \in T$ が存在しているのかに注目しているということである．

3.6 評価手法

得られた $C \in T$ の妥当性の評価をする必要がある．複数文書要約問題では，最も適切であると思われる重要文冗長性という指標を用いることができた．ほぼ同様の意味を表していて，要約を作成するにあたって冗長であるとされている文である．

評価を行う単位は 文レベルであって，文字通り任意の大きさの類似表現ではない．自由な大きさの同義表現に対応付けがなされたコーパスは存在しないようであり，人手による評価が最も適切であると思われるが，現時点まででは評価が行えてはいない．評価法の検討，実際の評価の実施も，今後の研究課題の一つである．

しかしながら，アラインメントの評価とは，具体的応用事例，アプリケーション，対象コーパスが定まっての評価となるため，必ずしも手法の良し悪しそのものを評価しているわけではない．本研究が提案する局所的類似性抽出技術は，個別アプリケーションの基盤技術として使用されることを目的としているため，処理効率，実行時間などで評価を行うことが重要であると思われる．あるいは，より上位のレベルでの評価，つまりシステム実装の容易さや，独立性などに意味があると考える．

3.7　本手法が対象とする問題

本技術の効果が期待できる問題の性質を以下に示す.

1. 対象は不特定多数の書き手による自由な自然言語による記述の集合である.
2. 対象の各要素（文書）には何らかのメタデータ，何らかの手がかりが付与されてはいない.
3. 対象の各要素は，その生成確率の独立性が平均的には高いと期待できる.
4. 対象の要素数は十分に多い.

一方，効果が期待できない問題を以下に示す.

1. 対象の記述には何らかの制約がある.
2. 対象の各要素は互いに依存性が高い.
3. 対象の各要素間の分散 (variance) が小さい.

具体的な問題に対して，例えばニュース記事や，その他のコンテンツの盗用（違法コピー）などの検出できるかということを考えるとする.

個別の問題に対しては，検出した局所的類似性をどのように用いればよいのか，の規定を行うことこそが要点であり，具体的に，どのような局所的類似性がどのように生じているのであれば盗用である（可能性が高い）と，設定できるのであれば検出ができるということになる．個別の具体的問題によって，類似性がどのような表現形として生じるのかは異なるため，実際のテキストデータの十分な観察に基づいた仕様を設定する必要がある.

第4章　コピー文字列検出に基づいたsplog filter

4.1　用語について

本稿中では，スパムコンテンツとされるブログエントリ (単一 URL が指し示す web ページ，ブログコンテンツ) のことを splog と表記し，そうではないブログエントリのことを blog と表記する．アルファベットで「blog」「splog」と記述している場合，それは コンテンツが splog であるか blog であるかを意味するラベルとして使う．splog か blog かを問わず任意のブログコンテンツを指す場合は「ブログ」と表記する．

4.2　splogの定義

本論文では，ブログは，人間が記述していると見なすことにする．これに反して，何らかの方法で他のコンテンツをコピーし自動的に記述されているブログエントリを，splog とする．

本論文では，佐藤ら [47] と同様に，splog を

> 一つ以上の何らかの他のコンテンツを部分的あるいは全てを
> コピーし，それらを連結して生成するブログエントリ

と，より一般性を持たせた考えでフィルタリング方法を構築する．

佐藤らの定義によるような場合分けを必要とせず，ほとんどの splog はこの 1 文で説明できるという点でが違いである．

splog 生成ツールが内部的に保持しているテンプレート，定型文，単語辞書など，明示的にその存在が明らかではない非公開のコンテンツを用いて作られた語やフレーズもコピーと考える．このようなテンプレートを使うこと自体が「コピー」であるからである.「今日は... について調べてみました！」や「... の最新ニュースです！」といったフレーズがテン

プレートの例にあたる．自動生成ツールは，その文章があたかも人間が書いたオリジナルコンテンツであるかのように見せかけるためにこのようなテンプレートを活用している．顔文字や慣用句など，一見しただけではわからない巧妙なテンプレートも数多く存在する．

著者が分類した splog の種類を表 4.1 , 表 4.2 にまとめる．各タイプの splog には，特に content snatch には数多くの亜種が存在しており，表 4.1 の template decorator や，combine の様な処理を施して生成されている場合が多い．

現在問題となっている splog とは，ほぼすべてがこのような自動生成ツールによって自動大量投稿されているものである．販売，流通されているツールも存在するのである．中には splogger が独自の改良を重ねた非公開の生成ツールなどによって生成されたらしき splog も存在する．

splog の定義をこのようにしたのは，自動生成ツールがこのような方法で生成しているからであり，そのような splog は，表 4.1 ,4.2 にまとめられるように，この定義に従って splog とみなすことができるからである．

人間が手動で他のコンテンツを引用した場合も splog となる可能性はある．オリジナルコンテンツが少しでもあれば，それは splog ではないが，コンテンツの全てが引用であれば，それは splog となる．ただし，前述したように，人間が手動で記述しているように見せかけるための巧妙なトリックが数多く用いられているため，それが本当にオリジナルコンテンツであるかどうか十分に注意して調査しなければならない．

splog 種別	説明文
search results	ある単語 (X) の検索結果コンテンツの 中身をコピーする. 単語 X は，自動投稿ツールが 持っている辞書の単語を使う場合と， 最近話題のキーワードを API で取得する場合とがある. 単語 X は複数の単語の場合もある
word salad	単語を無数に並べる. 自動投稿ツールが内部的に持っている 辞書の単語を使う場合と， 最近話題のキーワードを API で 取得する場合と， その両方を使う場合とがある.
template decorator	コンテンツに テンプレート文字列を付加する. 例えば「お早うございます」 「今日は……でした」「……をしています」 等
combine	表 4.1 , 表 4.2 の生成手法を 2 種類以上組み合わせる

表 4.1: splog 分類表 1

splog 種別	説明文
content snatch	他のコンテンツをコピーしてくる 以下のような 様々な亜種が存在する
news update	web ニュースの記事本文を すべてコピーして生成する. 記事本文に編集は全く加えない
mail magazine	メールマガジンの記事本文を すべてコピーして生成する. メール本文に編集は全く加えない
dictionary	wikipedia，などの百科事典コンテンツを そのままコピーして生成 する.
QA	yahoo 知恵袋，はてな人力検索，などの 質問文 (場合によっては回答文も) をそのままコピーする.
product induction	EC サイトの商品販売ページと 「機能レベルで」ほとんど同じであり そこから商品を直接購入もできる.
RSS	特定の RSS(Rich/RDF Site Summary， Really Simple Syndication) の内容をそのままコピーする 誰でもアクセス可能な RSS や sploggerが設定した RSS の場合もある

表 4.2: splog 分類表 2

図 4.1: product induction

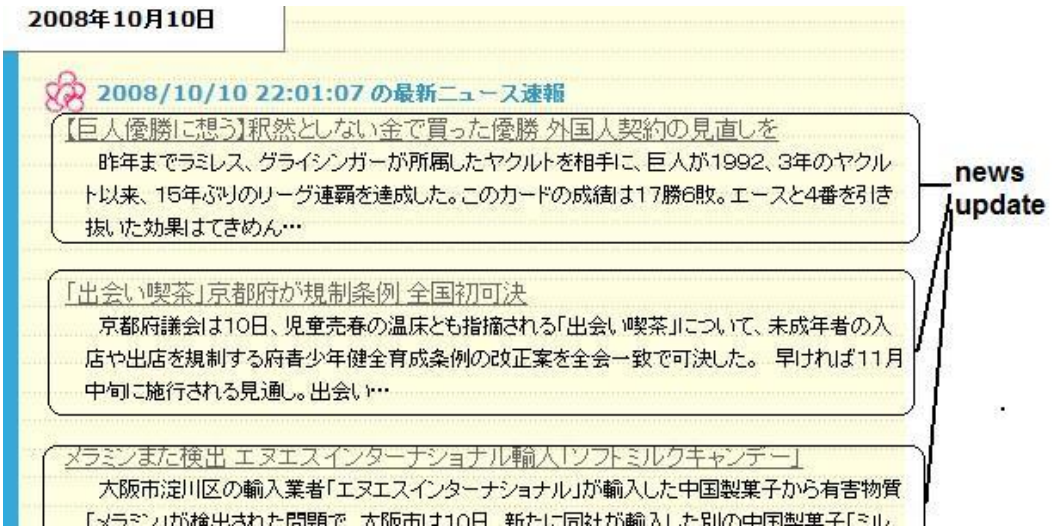

図 4.2: news update

図 4.1 に product induction の例を示す．EC サイトなどの商品の説明文をコピーし，販売の誘導を行う意図が明らかであり，splog であると非常にわかりやすい例であるが，このような例ばかりが splog ではない．

図 4.2 は news update combine の例である．これも，ニュース記事の冒頭とリンクを並べただけであり，splog であるという主張は受け入れ易い．しかし，これに template decorator が加わると，そうは思えなくなる人間は非常に多い．図 4.3 が，news update に template decorator を加えた例である．これは splog であるが，そう言われなければそうと気づかない人間は多い．ニュース記事本文の前後に テンプレート文字列を付加している．このようなテンプレートがあらかじめ用意されていて，その内の数種類をランダムに選択して追加している．

これがテンプレートである根拠であるが,（句読点，記号，全角，半角スペースまで）完全に同じ文字列を含んだブログエントリが他にも存在するからであり，しかも，完全にこのスタイル，この形式で存在しているからである．

図 4.4 が，word salad に template decorator を加えた例である．これも splog であるが，そう言われなければそうと気づかない人間は多い.「自動車」の部分には別の単語を入れても文章として成立するため，言葉を入れ替えれば大量にこのような文書を生成できる．もちろん，template の部分も，この 1 種類ではない．これは多数の template の中の 1 例であ

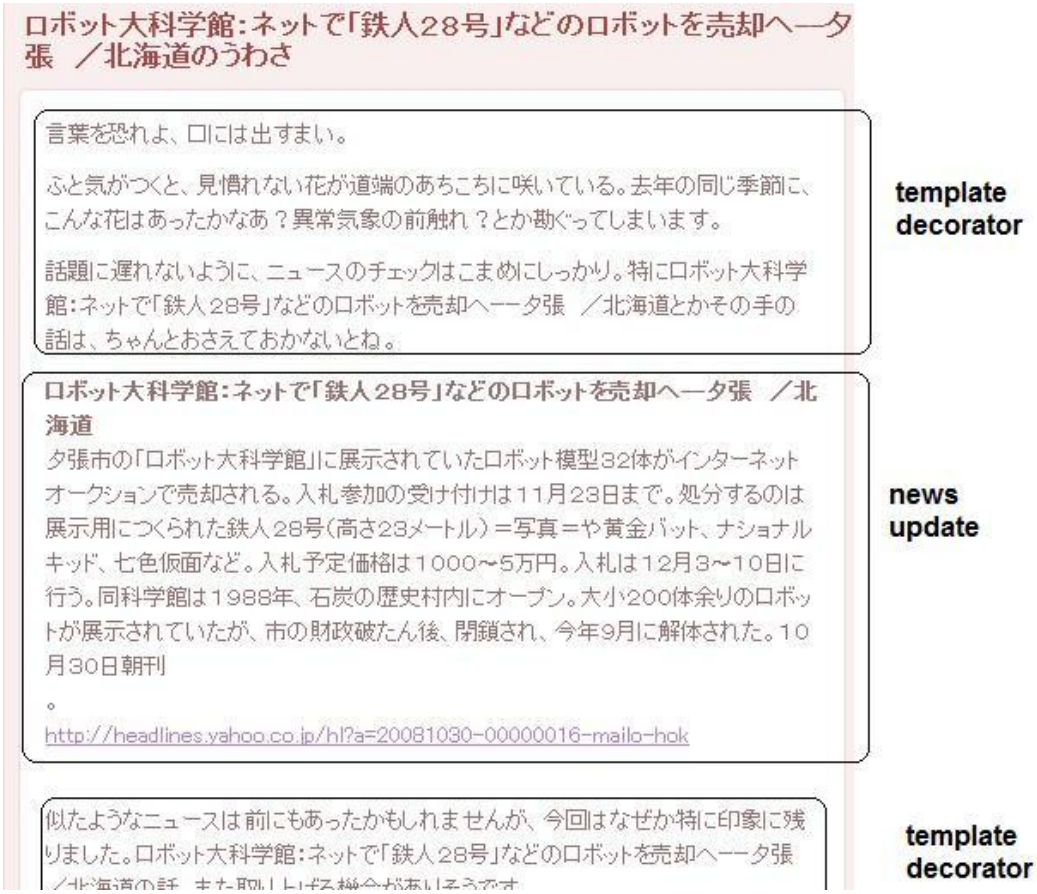

図 4.3: template decoretor 1

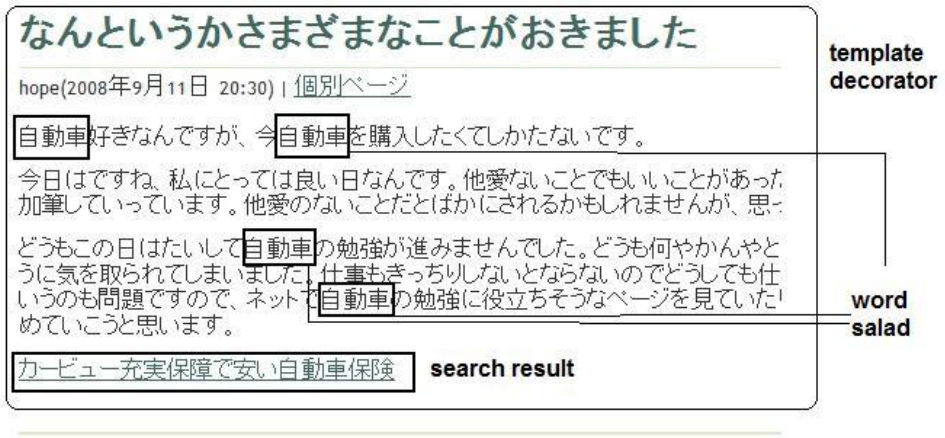

図 4.4: template decoretor 2

り，このテンプレートと単語の組み合わせを変えることで巧妙に気づかれないようにしている．

図 4.5 は最も気づかれにくい search results である．このブログエントリのタイトル中の「名古屋」「競馬」を検索クエリーとして検索をかけ，その検索結果の中から 1 文ずつ選択し，列挙しているだけである．

この文章が文章として成立しているようにも思える場合があるが，これは無作為にただ列挙しているだけである．例えば，それぞれの文の順番を変えても，文章として成り立っていると思える場合もある．文章を読む人間が文間の情報を補完することによって，人間が記述している”自然な文”であるという錯覚を起こしている．

これが検索結果であり自動生成コンテンツであるとする根拠は，1 文目の 「Permalink | トラックバック (0) | 19:27」である．このような文字列はブログエントリの本文中ではなく，フッターとして表れ，ユーザーが明にコンテンツと意識する部分には基本的には使われることはない．このような文字列は検索結果には現れることは多いが，このブログエントリ中でこのような文字列を記述する文脈上の理由もない．

本エントリだけを単独で見ていてもそのような不自然さを指摘することはできるが，このユーザが投稿している他のブログエントリも調査すればその不自然さは明らかである．このユーザの場合,「名古屋」に他の検索クエリーをランダムに加えてその検索結果から splog を生成している．

単一ブログエントリの不自然さだけではなく，周辺の情報まであわせ

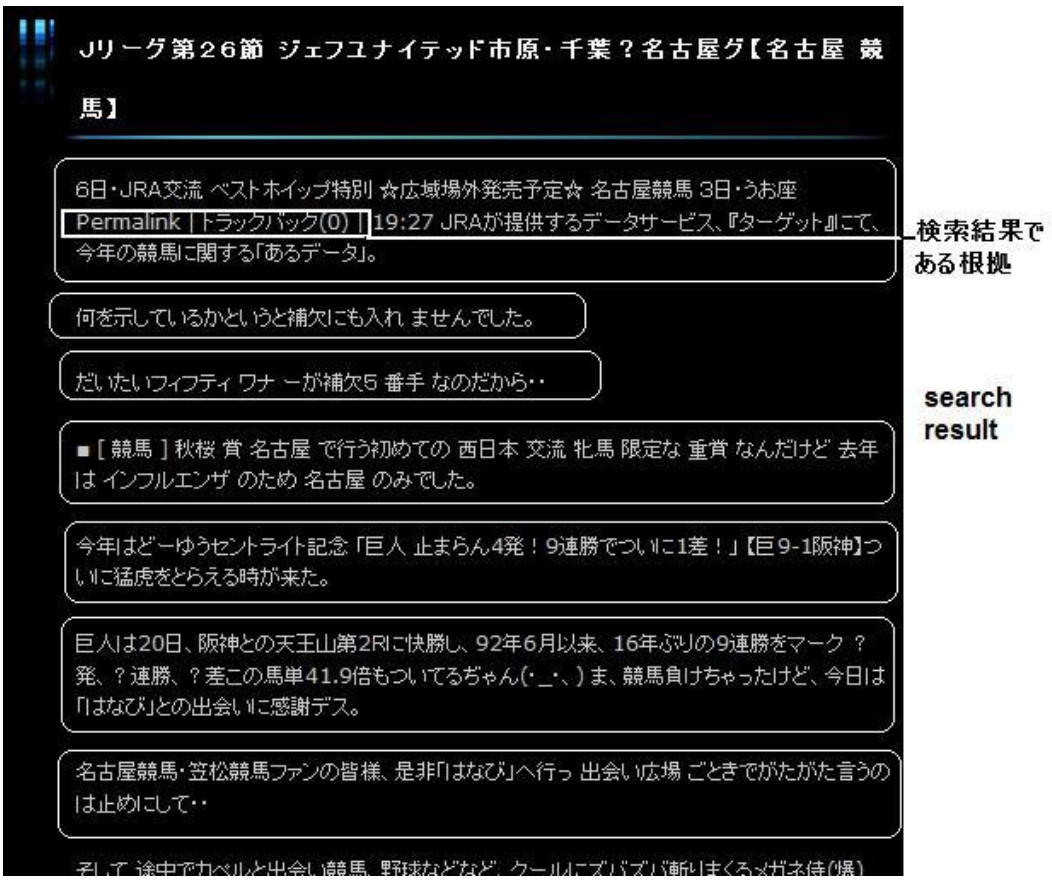

図 4.5: search result combine

てそれが不自然なコンテンツでないかどうかを調べることによって，本調査ではできるだけ正確に splog を特定した．今回のコーパス中にも表 4.1 の template decorator に該当するようなテンプレートと思われる不自然に長い文字列の完全一致が頻繁にあり，このようなテンプレートの検出 [51] からも splog の検出を行うことができると期待できる．

佐藤ら [47] は，特定のキーワードを含むブログは splog である確率が 40%以上という報告をしたが，本稿ではこれとは異なる主張を行う．

splog テンプレート集合 *template* に該当する長いフレーズ $sentence_x \in template$ ($sentence_x$ は複数文にわたる文字列の場合もある)，が存在するブログエントリは高い確率で splog である．

$sentence_x$ の例として，”私の資料室・みんなの資料広場 HAPPYCAMPUS”という文字列がある．今回構築したコーパス中にはこの文字列が入ったブログエントリが 72 あり，その全てが splog であった．これは splog に顕著に現れるテンプレート *template* の一つである．

splog テンプレート集合 *template* は，そのような ”実例 ”の集合であり，その検出方法などは研究の余地がある．

4.3 splog フィルタリング手法

4.3.1 記号の定義

まず，本稿で用いる記号を定義する．文字列 $\boldsymbol{s}$ の i 番目の文字を s_i と表す．文字列 $\boldsymbol{s}$ の 長さ i の接頭辞を $\boldsymbol{s}_{:i}$ と，また，$\boldsymbol{s}$ の i 番目の文字から始まる接尾辞を $\boldsymbol{s}_{i:}$ と表す．$\boldsymbol{s}_{i:j}$ は，$\boldsymbol{s}$ の i 番目から j 番目の部分文字列を表す．$|\boldsymbol{s}|$ は，$\boldsymbol{s}$ の長さを表す．

定義 4.2 から，splog 中の文字列は他のコンテンツにも出現していると思われる．この前提を元に，ブログの copy rate を定義する．copy rate は，そのブログ中の部分文字列がどの程度他のコンテンツに存在しているかを示す値であり，提案手法はこの値に基づいて splog 判定を行う．

4.3.2 コピー検出法

単語レベルの短い文字列は blog にも頻繁に表れるため，そのような短い領域もコピーとみなすことは本稿の目的には合致しない．一方，一文以上にわたる長い文字列が一致する場合は，コピーされている可能性が

高いと思われる．そこで，提案手法では，一定長以上の文字列が複数ブログに重複して出現する場合，その文字列はコピーされたとみなすことにする．この文字列長の下限を l とする．

コピー検出のために用意されたブログの集合を B とする．splog 判定を行うブログ $\boldsymbol{b}$ の任意の部分文字列 $\boldsymbol{s}$ に対して，文字列 $\boldsymbol{s}$ を含む B 中のブログ数を $df(\boldsymbol{s})$ で表すことにする．このとき，文字列 $\boldsymbol{s}$ のコピー長を以下のように定義する．

$$cpl(\boldsymbol{s}) \equiv \begin{cases} |\boldsymbol{s}| & |\boldsymbol{s}| \geq l, df(\boldsymbol{s}) \geq 2 \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

式 (4.1) で定義されるコピー長は，文字列の長さをそのまま用いているが，重み付きの文字列長を用いることもできる．たとえば，情報検索などで用いられる IDF(Inverse Document Frequency) をコピー文字列長として用いると，以下のような重みつきコピー文字列長が得られる．

$$cpl(\boldsymbol{s}) = \begin{cases} \log \frac{|B|}{df(\boldsymbol{s})} & |\boldsymbol{s}| \geq l, df(\boldsymbol{s}) \geq 2 \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

次にブログコンテンツのコピー長を定義する．以下では，ブログコンテンツを文字列として扱う．まず，ブログコンテンツ $\boldsymbol{b} = c_1 c_2 \cdots c_l$ を，以下に示すように n 個の部分文字列に分割することを考える．

$$\underbrace{c_1 \cdots c_{p_1}}_{\boldsymbol{b}_1} \underbrace{c_{p_1+1} \cdots c_{p_2}}_{\boldsymbol{b}_2} \cdots \underbrace{c_{p_{n-1}+1} \cdots c_l}_{\boldsymbol{b}_n} \tag{4.3}$$

以下では，このような分割を，分割によって得られる部分文字列のリスト

$$\boldsymbol{p} \equiv (\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots \boldsymbol{b}_n) \tag{4.4}$$

で表すことにする．

ブログコンテンツ $\boldsymbol{b}$ の分割 $\boldsymbol{p}$ によるコピー文字列長を以下に示すように，$\boldsymbol{p}$ によって得られる部分文字列のコピー文字列長の和と定義する．

$$bl(\boldsymbol{b}, \boldsymbol{p}) \equiv \sum_{i=1}^{n} cpl(\boldsymbol{b}_i) \tag{4.5}$$

部分文字列のコピー文字列長 $cpl(\boldsymbol{b}_i)$ としては，(4.1) 式や (4.2) など，問題に応じてコピー文字列長を定義し直すことで文字の並び順を考慮する必要がある様々な問題に適応できる．本稿では，式 (4.2) を用いる．

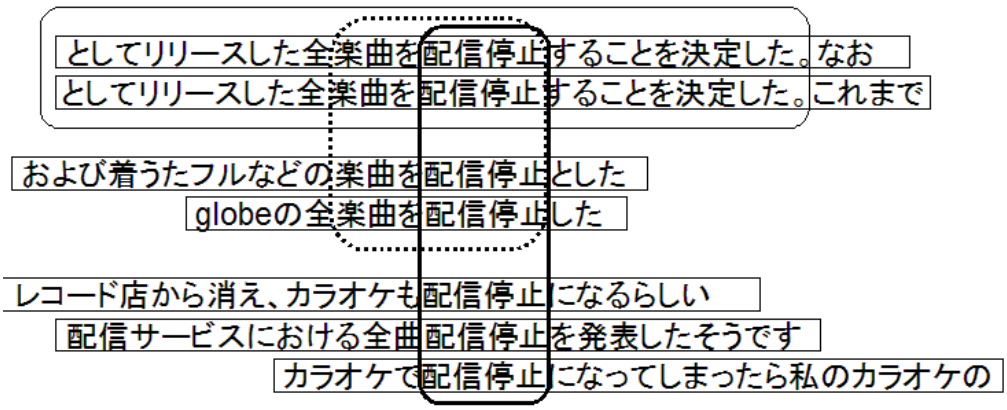

図 4.6: 一致分割の例

最後に，ブログコンテンツのコピー文字列長を，分割によって得られるコピー文字列長の最大値と定義する．

$$bl^*(\boldsymbol{b}) \equiv \max_{\boldsymbol{p} \in P(\boldsymbol{b})} bl(\boldsymbol{b}, \boldsymbol{p}) \tag{4.6}$$

ここで，$P(\boldsymbol{b})$ は，ブログコンテンツ $\boldsymbol{b}$ の可能な分割の集合を表している．図 4.6 の例では次のような式が成り立つ．

df(としてリリースした全楽曲を配信停止することを決定した.)
$< df$(楽曲を配信停止)
$< df$(配信停止)

このように，すべての部分文字列の一致を検出し，最大の値をとる分割を探し出す式である．

4.3.3 コピー文字列長計算アルゴリズム

文字列 $\boldsymbol{s}$ の可能な分割は，$O(2^{|\boldsymbol{s}|})$ の数だけあるため，全ての分割を列挙して，式 (4.6) を解くことは計算時間の観点から現実的でない．文字列 $\boldsymbol{s}$ の任意の接頭辞 $\boldsymbol{s}_p$ に対して残りの接尾辞を $\boldsymbol{s}_s$ と表すことにする．すると，(4.6) は以下のように再帰的に表すことができる．

$$bl^*(\boldsymbol{s}) = \begin{cases} 0 & |\boldsymbol{s}| < l \\ \max_{\boldsymbol{s}_s} \{bl^*(\boldsymbol{s}_p) + cpl(\boldsymbol{s}_s)\} & \text{otherwise} \end{cases} \tag{4.7}$$

ここで，最大値は，ブログコンテンツ $\boldsymbol{s}$ の任意の接頭辞に対して求めるものとする．

(4.7) 式に基づいて，コピー文字列長を計算するためには，ブログコンテンツの任意の部分文字列 $\boldsymbol{s}$ に対して，その文書出現頻度 $df(\boldsymbol{s})$ を求める必要がある．$df(\boldsymbol{s})$ は，suffix array を用いることで効率的に求めることができる．suffix array [20] は，コピー文字列の検出を行うための文書集合 B 中のすべての接尾辞を辞書順にソートし，接尾辞へのポインタを保持している．そのため，文字列 $\boldsymbol{s}$ を接頭辞として含む B 中の接尾辞は，array 上で連続した領域に表れる．suffix array 上のこの領域の先頭および末尾の位置をそれぞれ $s(\boldsymbol{s})$ および $e(\boldsymbol{s})$ と表すことにする．この領域の中の接尾辞を含むブログエントリを数えることによって，$df(\boldsymbol{s})$ を求めることができる．具体的には，suffix array を 2 分探索し，$\boldsymbol{s}$ を接頭辞として含む接尾辞をひとつみつけ，さらに suffix array 上の前方および後方を走査して，$\boldsymbol{s}$ を接頭辞に含む領域を見つける．suffix array の 2 分探索には $O(\log n)$，走査には $e(\boldsymbol{s}) - s(\boldsymbol{s}) + 3$ の文字列比較が必要になるが，$df(\boldsymbol{s})$ が大きい場合には，suffix array の走査の時間が支配的となる．

Suffix array の性質から，文字列 $\boldsymbol{t}$ が $\boldsymbol{s}$ の接頭辞の場合，以下の関係が成り立つ．

$$s(\boldsymbol{t}) < s(\boldsymbol{s}) < e(\boldsymbol{s}) < e(\boldsymbol{t})$$

そこで，(4.7) の最大値を求める際に，まず，最も長い接尾辞 $\boldsymbol{s}$ に対して suffix array 上の領域 $(s(\boldsymbol{s}), e(\boldsymbol{s})$ を求め，その接頭辞に対して，順次領域を前後に広げる．これにより，ブログコンテンツの各接尾辞に対して，1 回の suffix array の 2 分探索と $\boldsymbol{s}_{1:l}$ に対応する領域を一度走査することによって，最大値を求めることができる．

図 4.7 にブログコンテンツのコピー文字列長を求めるアルゴリズムを示す．$s(\boldsymbol{b}_{j:i})$ および $e(\boldsymbol{b}_{j:i})$ は，それぞれ，suffix array A を 2 分探索し，部分文字列 $\boldsymbol{b}_{j:i}$ の A 上の開始位置と終了位置を求める関数を，また，$f(s, e)$ は，A 上の位置 s および e の間に含まれるブログ数を求める関数を表している．

$s(\boldsymbol{b}_{j:i})$ と $e(\boldsymbol{b}_{j:i})$ を suffix arrays 中から見つけるための計算量は $O(log|A|)$ であり，ブログコンテンツ b 全体でこれを $O(log|\boldsymbol{b}|^2)$ 回実行することになる．コピー長計算には，各々 $O(log|A| \cdot log|\boldsymbol{b}|^2)$ の計算量を要する．

suffix array の構築に要した処理の時間のグラフ 図 4.12， 4.13 に示す．

なお，出現頻度の高い部分文字列については，毎回，suffix array 上を走査し，その部分文字列を含むブログ数を計数するコストが高いため，現

input: ブログコンテンツ $\boldsymbol{b}$, 最小コピー文字列長 l
  suffix array A
output: ブログコンテンツ $\boldsymbol{b}$ のコピー文字列長

begin
  set 0 to all components of C
  for $i = l + 1$ to $|\boldsymbol{b}|$ **do**
    for $j = 1$ to $i - l$
      $s \leftarrow s(\boldsymbol{b}_{j:i}, A)$, $e \leftarrow e(\boldsymbol{b}_{j:i}, A)$
      $C[i] \leftarrow max(C[j], cpl(\boldsymbol{s}, f(s, e)) + C[j])$
    end
  end
  return $C[|b|]$
end

図 4.7: コピー長計算アルゴリズム

在の実装では，関係データベースに部分文字列とその頻度を登録して用いている．

4.4 評価用データ

4.4.1 作成方法

本研究で提案するフィルタリング法を評価するために，splog ベンチマークデータを作成した．データは，2008 年 1 月 31 日 18:00 から 2008 年 2 月 5 日 06:00 までの間，日本国内の大手 blog CSP(Content Service Provider) のブログからサンプリングした．各社が配信している新着ブログエントリの RSS フィードを常時監視し，サンプリング率 0.01 でランダムサンプリングを行った．繰り返しになるが，本研究では特定のタイプの splog を実験的に作成したわけではない．また，特定の検索クエリーに対する検索エンジンの検索結果を集めたりしたなど，統計的傾向を崩したわけではない．表 4.3 に各社別 blog,splog の実数値を示す．これをグラフにしたものが図 4.8 である．

4.4.2 Labeled entries

収集したブログ（webページを）を目視しsplog/blogの判定を行い，計21,668件のラベル付きリストを作成した．まず，表4.1，表4.2に示すようにsplogを分類し，それぞれの具体例を準備した．次に，3名の作業者に分類と例を提示し，splog判定を行わせた．判断がつかない場合は，同一bloggerの過去のブログエントリまで確認し，不自然な点がないかを確認した上でラベリングを行った．表4.3に，得られたラベル付きリストのCSP別のblog，splog数を示す．

splogは一見しただけでは自動生成コンテンツであるとは気づかれないように作られている．しかし，その特徴的傾向に注目すれば明らかに不自然な点がわかるようになる．その結果，過去のブログエントリを確認する必要なく，単一ブログエントリのテキストだけから判断がつくようになった．このようにして著者が作業者のラベリング結果を添削し，作業者のsplog検出を教育しつつラベリング作業を行った．

4.4.3 Unlabel entries

この21,668件以外に，splog/blogラベルが付いていない同期間のブログコンテンツを含め50,000件収集し，実験に用いた．

4.4.4 Search API

データ収集期間に話題となった語をサーチエンジンより収集した．単語の収集を行った期間は2007年10月1日から2008年2月6日までである．使用したサーチエンジンは以下のとおりである．

1. http://kizasi.jp/
2. Yahoo!検索ランキング
   http://searchranking.yahoo.co.jp/
3. テクノラティジャパン：人気のブログ検索キーワード
   http://feeds.technorati.jp/trjcf/keyword_ranking/
4. gooキーワードランキング
   http://ranking.goo.ne.jp/keyword/

5. 注目キーワード はてなダイアリー
   http://d.hatena.ne.jp/hotkeyword

こうして収集した単語 (13,733 語) を検索クエリーとして検索 API で検索結果を収集した．検索を行った期間は 2008 年 2 月 6 日から 2008 年 2 月 8 日である．使用した検索 API は以下のとおりである．

1. livedoor ブログ検索
2. goo ブログ検索
3. Nifty @search
4. Namaan
5. Yahoo ブログ検索
6. Technorati ブログ検索
7. Google ブログ検索
8. エキサイトブログ
9. So-net ブログ検索

検索結果として最初のページに表示されるページのスニペットを収集した．検索結果として表示される部分，つまり，API が返す XML のうち，<ITEM> 中の，<description> の中身を収集する．検索クエリーによっては，検索結果が標準の検索結果表示数に満たないことが多かったため，返ってくる件数は一定ではなかった．

このようにして，検索結果約 100 万件分のコーパスを作成した．ただし，この内容には重複が非常に多くある．異なる検索エンジンで同一ページを表示することは容易に想定できるし，異なる検索クエリーに対しても同一ページを検索結果の上位に表示することは考えられるからである．

このような方法でコーパスを構築する理由は，splog 生成ツールが，このような方法で splog を生成していると予想されるからである．表 4.1 の splog の種類のうち，word salad， search result などはこのような手法で生成を行っていると思われるため，元となる文書を収集することでコピー検知を行う．

こうして作成したベンチマークテストデータを用いて，提案するフィルタリング手法で blog/splog 判定を行う．今回はデータとしてブログエントリの本文のみを対象とする．

CSP	blog	splog
livedoor Blog	2208	936
goo ブログ	1297	119
LOVELOG	68	8
Yahoo!ブログ	1935	59
アメーバブログ	4029	306
JUGEM	1144	523
ココログ	713	341
FC2 ブログ	762	1155
ヤプログ！	1616	40
Seesaa ブログ	148	1065
はてなダイアリー	273	14
ウェブリブログ	325	19
teacup. ブログ	484	48
So-net blog	244	93
忍者ブログ	447	41
ドリコムブログ	45	0
楽天広場	950	98
AOL ダイアリー	44	1
Iza!	61	5
CURURU	3	0
Cnet	1	0
計	16797	4871

表 4.3: CSP 別 splog 含有率

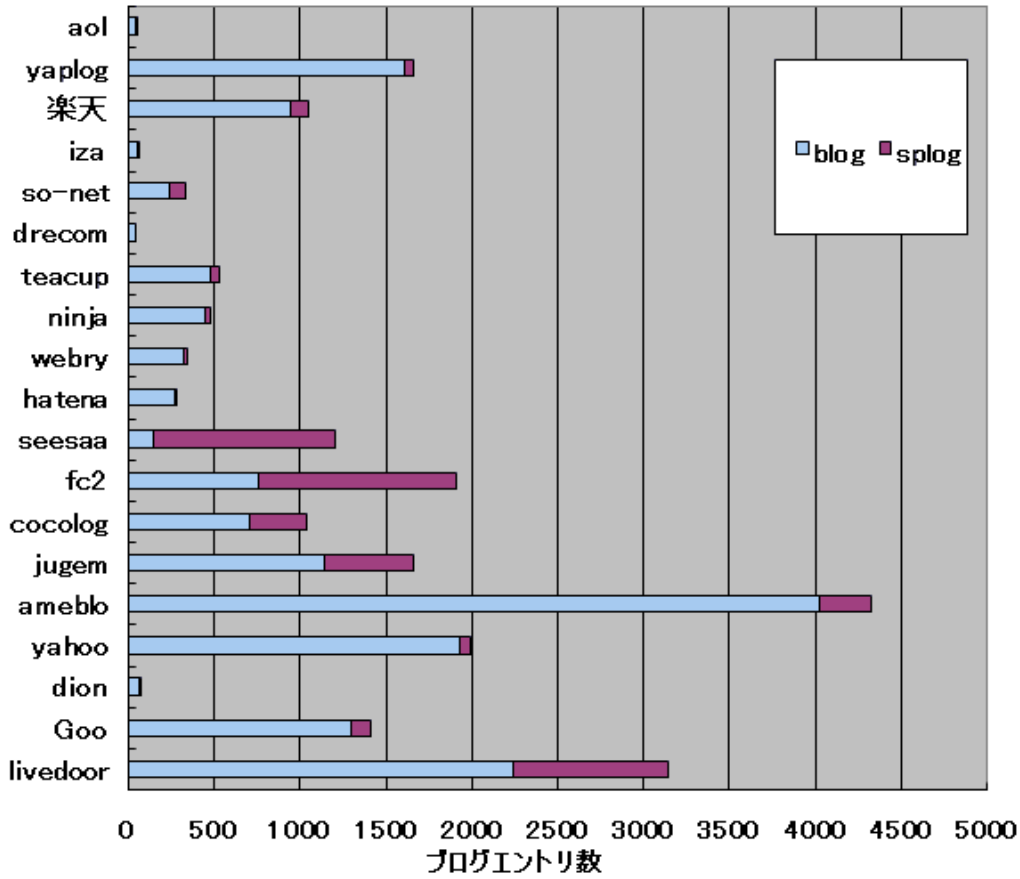

図 4.8: CSP 別 splog 含有率

4.5 誤差範囲

このデータセットが実際の blogosphere の縮図としての信頼度がどの程度あるのか大標本法による母比率 $\hat{p}$ の推定を行う．必要なサンプル数を算出する根拠となっている数式は次の通りである．

$$\frac{N}{(\frac{E}{k})^2 \frac{N-1}{P(100-P)} + 1} \tag{4.8}$$

- N: 母集団の大きさ
- E: 許容できる誤差の範囲
- P: 想定する調査結果
- k: 信頼度係数

サンプルの値は 0 か 1 しかとらないものとすると，サンプルの値の平均値が サンプルが 1 の確率に等しくなるため，以下の式が成り立つ．

$$\begin{aligned} \text{標本の大きさ} &: n \\ \text{標本平均} &: x = E(x_a) \\ \text{母集団の標準偏差} &: \sigma \\ &= \sqrt{\frac{1}{n}}\sqrt{E(x_a^2) - E(x_a)^2} \\ &= \sqrt{\frac{1}{n}}\sqrt{1^2 x - x^2} \end{aligned}$$

$$(E(x_a^2) = \frac{(1)^2 \times n \times x + (0)^2 \times n \times (1-x)}{n} = 1^2 x) \tag{4.9}$$

$$x - \text{t 値}\frac{\sigma}{\sqrt{n}} \leq \hat{p} \leq x + \text{t 値}\frac{\sigma}{\sqrt{n}} \tag{4.10}$$

例えば，全体の splog 率は $\frac{4871}{21668} = 0.2248$ である．これが，サンプルが splog である確率であり，標本平均の値である．ここから標準偏差が計算できる

$$\sigma = \sqrt{0.2248 - 0.2248^2} = 0.41745 \tag{4.11}$$

信頼度 95%の場合 t 値は 1.96 である．このとき splog 率の信頼区間は以下の範囲になる．

$$0.2220 \leq \text{全体のsplog 率} \leq 0.2276 \tag{4.12}$$

同様に，例えば livedoor blog が日本語ブログに占める割合の 95%の信頼区間は以下の範囲になる

$$0.14271 \leq \text{livedoor blog の割合} \leq 0.14749 \tag{4.13}$$

4.6 ブログの本文抽出

ブログ中の広告などを除いて本文のみを抽出するために，html のタグを用いた．ブログ本文を表すタグは CSP ごとに異なるため，文献 [34] と同様に，それぞれの CSP の本文タグを調べ，その中のテキストだけを取得することにより本文を抽出した．

4.7 実験結果

4.7.1 実験の概要

4.4 章で述べたラベル付きブログエントリすべてに対して，4.3 章で提案したアルゴリズムを用いて各ブログエントリのコピー文字列長を計算し，これに基づいて splog の判定を行った．つまり，コピー文字列長が閾値 *threshold* 以上のブログエントリを splog とみなし，*threshold* 未満の場合は blog とみなした．

コピー文字列を検知するためのデータセットとして 4.4 章の述べたコーパスそれぞれを用いた場合の結果を調べた．

- Labeled entries
- Unlabel entries
- Search API
- Unlabel entries+Search API

提案手法では，最小コピー文字列長を決める必要がある．実験では，このパラメタの値として $1, 3, 5, 7, 10, 12, 15, 20, 25$ を用いた．

4.7.2 評価指標

splog フィルタリングの性能をはかる評価指標として，以下に示す F 値を用いた．

$$F\text{ 値} \equiv \frac{2\,r\,p}{r+p} \tag{4.14}$$

ここで，r および p は，以下に示す recall と precision を表している．

$$r \equiv \frac{\text{システム出力と正解ラベルが splog であると一致する数}}{\text{データ中の splog 数}} \tag{4.15}$$

$$p \equiv \frac{\text{システム出力と正解ラベルが splog であると一致する数}}{\text{システムが splog であると出力した数}} \tag{4.16}$$

閾値を変化させることで precision ，recall はそれぞれ変わり，precision-recall 曲線を描くことができるが，今回は，各コーパス別に F 値が最大となる閾値の時の precision ， recall を記録し，比較を行う．

4.7.3 実験結果と考察

図 4.9 に，最小コピー文字列長とフィルタリング性能の関係を示す．コピー文字列を検知するためのデータベースとして，4.7.1 節に述べた 4 種類のコーパスを用いた．グラフ中の F 値は，21,668 件のラベル付きブログエントリの F 値の平均値を表している．

図 4.9 に示されているように，最小コピー文字列長が短い場合，フィルタリングの性能は大きく劣化する．この理由として，短い単語やフレーズは blog でも出現頻度が高いが，最小コピー文字列長が短い場合は，このような頻出単語やフレーズもコピー文字列とみなしてしまうことがあげられる．そのため，recall の上昇を超える precision の減少が生じ，F 値が下がってしまうものと思われる．

逆に最小コピー文字列長を 15 以上に設定した場合も，わずかではあるがフィルタリング性能が劣化している．これは，最小コピー文字列長以下のコピー文字列の検知ができなくなってしまうことに起因する．最小コピー文字列長を長くすることによって，precision を向上させる効果を得られるが，recall がそれ以上に減少してしまい，結果として F 値を下げてしまうものと思われる．

Labeled entries				
l	threshold	precision	recall	F
1	1600	0.525	0.524	0.524
3	1150	0.524	0.529	0.526
5	540	0.546	0.545	0.546
7	200	0.601	0.592	0.596
10	40	0.688	0.699	0.693
12	19	0.748	0.682	0.713
15	9.3	0.752	0.648	0.696
20	0.01	0.735	0.648	0.689
25	0.01	0.816	0.558	0.663

表 4.4: Labeled entries

Unlabel entries				
l	threshold	precision	recall	F
1	1630	0.521	0.523	0.522
3	1250	0.524	0.529	0.526
5	410	0.572	0.578	0.575
7	165	0.645	0.648	0.646
10	65	0.718	0.703	0.710
12	29	0.733	0.729	0.731
15	10.5	0.743	0.722	0.732
20	8.5	0.730	0.704	0.717
25	0.01	0.809	0.639	0.714

表 4.5: Unlabel entries

Search API				
l	threshold	precision	recall	F
1	1480	0.529	0.529	0.529
3	1265	0.520	0.515	0.517
5	650	0.549	0.556	0.553
7	180	0.650	0.645	0.648
10	69	0.718	0.711	0.715
12	29	0.731	0.740	0.736
15	13	0.735	0.746	0.740
20	7	0.724	0.731	0.727
25	0.01	0.809	0.658	0.726

表 4.6: Search API

Unlabel entries+Search API				
l	threshold	precision	recall	F
1	1400	0.511	0.514	0.513
3	1115	0.516	0.514	0.515
5	800	0.534	0.547	0.541
7	385	0.601	0.602	0.601
10	107	0.711	0.713	0.712
12	50	0.745	0.740	0.743
15	21.2	0.757	0.751	0.754
20	10.8	0.792	0.711	0.750
25	0.01	0.772	0.731	0.751

表 4.7: Unlabel entries+Search API

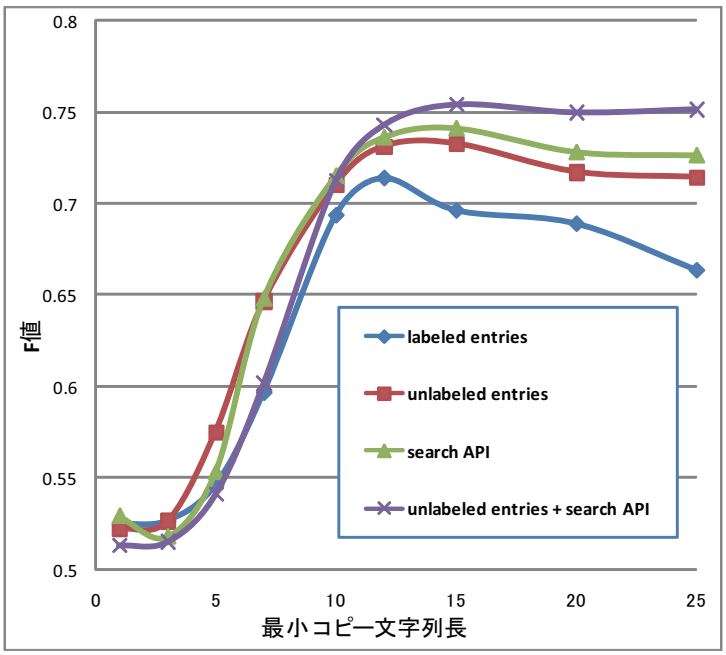

図 4.9: 最小コピー文字列長とフィルタリング性能

コーパスによる影響であるが，図 4.9 に示されるように，“Labeled entries” よりも”Unlabel entries” や”Search API” の方が平均的に少し高い精度を示している．また “Unlabel entries” よりも “Search API” の方がわずかに精度が上である．しかし，今回の実験では，コーパスによるフィルタリング性能の変化はほとんど観測されなかった．

一般的には *threshold* を大きく設定するほど precision を高くでき，正確に splog を検出できるようになる．顕著な splog にはそれだけコピー文字列が多いということを意味している．しかし同時に recall が低下することから，見かけ上のコピーが少ない splog も多く存在すると思われる．

図 4.9 の各点における precision，recall の値を表 4.4 4.5 4.6 4.7 に示す．コピー文字列長が大きい場合，コピー率の値の大小はほとんど意味がなくなる．l が 25 の場合には，どのコーパスでも threshold は 0.01 と設定する．コピー率が 0 でなければ splog と判定することで，splog 検出の precision は非常に高くできることを示している．これは，十分に長い文字列の一致が存在する場合は，その一致を含む全てのブログエントリが splog の可能性が高いということを意味している．これは必ずしも直感的ではない場合があるが，句読点，改行コード，タブ，記号なども含め

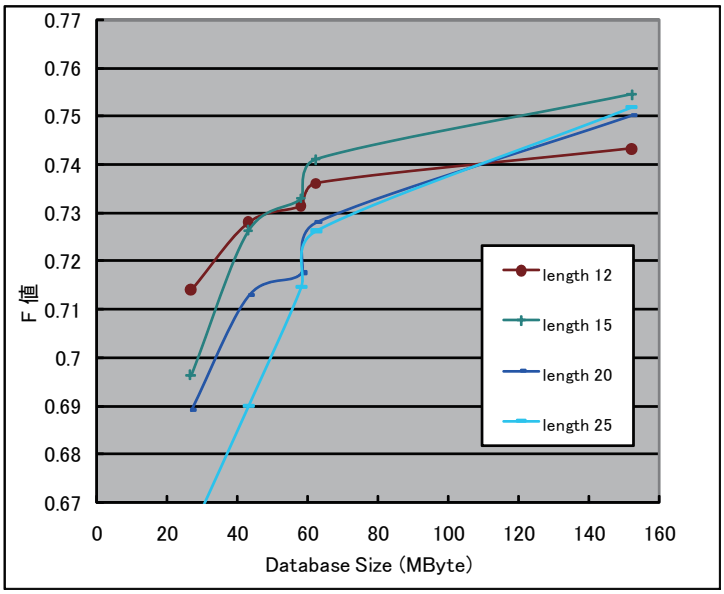

図 4.10: データベースサイズとフィルタリング性能

た完全一致であり，例えば 30 文字以上の一致などは，コピーしたのでなければ，自然（偶然）にはほとんど起こり得ない．そして，splog にはそのような”異常”な文字列が高い確率で存在しているということを意味している．

表 4.8，図 4.10 には，コピー検知に用いるデータの大きさとフィルタリング性能の関係を示す．図は，それぞれ最小コピー文字列長が 12,15,20,25 の場合の性能の変化を示している．

コーパスを大規模にすることが splog フィルタリングに有効であるが [34]，今回の実験では，コーパスの規模が，60MB を越えたあたりから性能向上の鈍化がみられた．頻度の高いコピー文字列は比較的小規模のコーパスでも検知することができる．一方で，頻度の低いコピー文字列を検出するためには，コーパスサイズを増加させるとともに，コピー文字列検出に効果的な情報源をみつける必要がある．

60MBytes 付近でグラフにずれが生じている理由はコピー検知のためのコーパスが Unlabel entries と Search API　とで異なるためである．この付近にある程度の差があるが，コーパスの内容とは関係なくデータ量に対して F 値が単調増加することは非常に興味深い結果である．これは，コピー検知のための情報源がどのようなものであるかは重要ではないと

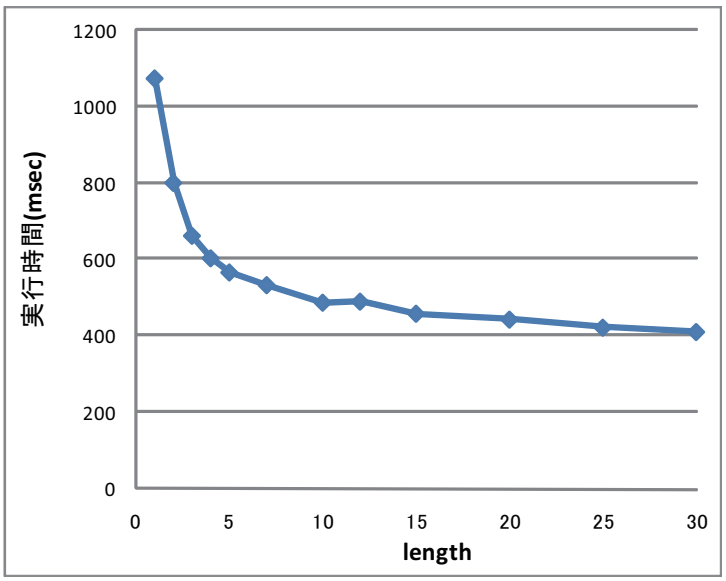

図 4.11: 最小コピー文字列長と処理時間

いうことを示している可能性がある．

4.7.4 処理性能と考察

提案手法の処理性能を調べるために，さまざまな最小コピー文字列長およびコピー検知用のデータベースサイズに対する，21,668 件のラベル付きブログエントリの処理時間を計測した．

表 4.9，図 4.11 は，最小コピー文字列長を変化させたときの 1 ブログエントリあたりの平均実行時間を示している．このグラフに示されるように，最小コピー文字列長が短いほど処理時間が長くなる．

これは，式 (4.1) で示す一致長の下限による判定の計算量が急激に増加するためである．短い文字列を接頭辞とする接尾辞は多数あるため，suffix array 上で文字列を接頭辞として持つ接尾辞の範囲を検索した後に，その範囲の接尾辞を走査して出現頻度を計数する時間が増加するためである．

図 4.14 は，最小コピー文字列長が 1 で処理時間が最もかかる場合の各ブログエントリの処理時間を示している．4.3 章で提案したアルゴリズムは，ブログエントリ長 n に対して，suffix array を $O(n^2)$ 回検索する必要があるが，図 4.14 は 処理時間が $O(n)$ に近いことを示している．すでに述べたとおり，短い文字列の出現頻度を求めるためには時間がかかるが，

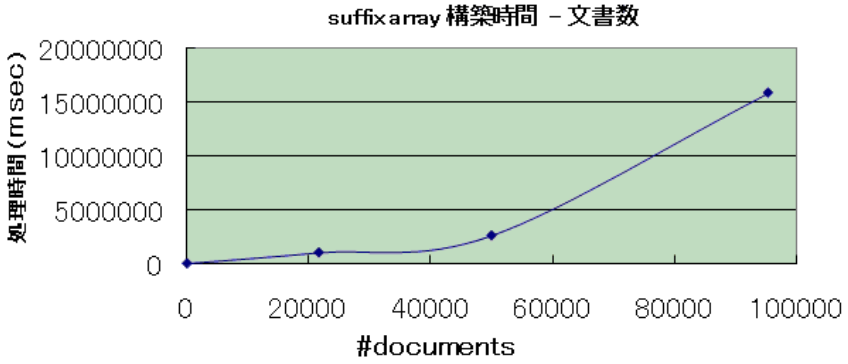

図 4.12: Suffix Array 構築時間-文書数

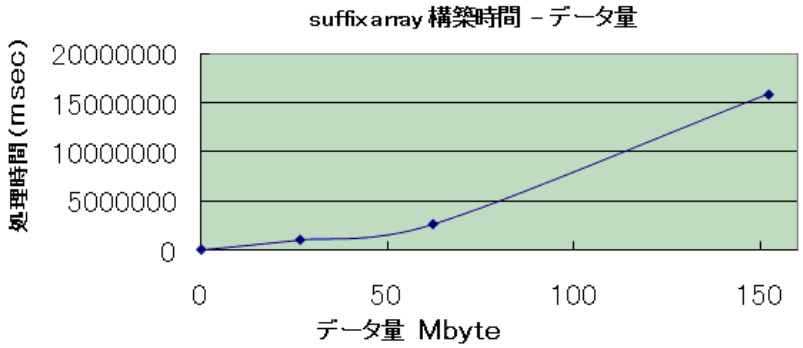

図 4.13: Suffix Array 構築時間-データ量

一方，長い文字列を接頭辞とする接尾辞は少なく，高速に出現頻度を求めることができる．例えば，ブログエントリ中で，長さ H 以下の部分文字列の総数は，ブログエントリの長さに対してはほぼ線形に比例する．このため，ブログエントリ長に対するコストの増加量は線形に近いものとなるためである．

4.7.5 スパムテンプレート検出法との比較

splog フィルタリングには，splog のさまざまな特徴を用いることが考えられる．例えば，平均的に blog よりも splog の方がコンテンツ長が長く，本研究で作成したコーパスでは，blog の平均の長さは 488.299 文字に対して，splog の平均の長さは 2263.331 文字であった．

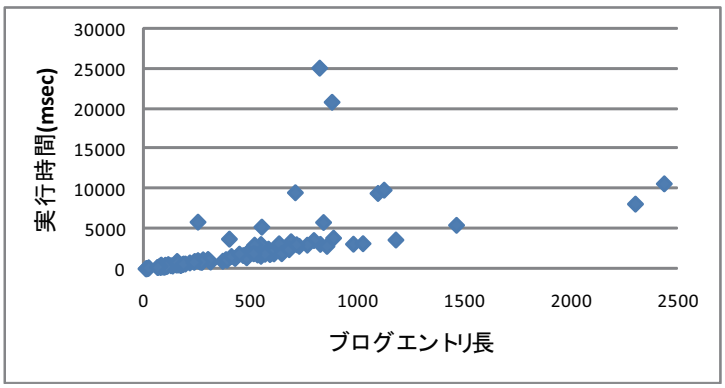

図 4.14: ブログエントリの長さと処理時間

そこで，ブログのコンテンツ長を特徴量とするフィルタリング法との比較を試みた．しかし，この方法のフィルタリング性能は，提案手法よりもかなり劣っていた．そこで，本論文では，成澤ら [51] が提案したスパムテンプレート検出法を用いたフィルタリング法との比較を行う．まず，文献 [51] の方法にもとづいて，下記の手順でスパムテンプレートを検出する．

1. コーパスの Suffix Array を構築し部分文字列の数え上げによってジップ則に反する最も出現頻度の高い文字列を検出する．この出現頻度を f_s とする．

2. 出現頻度 f_s の部分文字列の中で最長の文字列を検出する．この文字列を $\boldsymbol{s}$ とする

3. $\boldsymbol{s}$ をコーパスから除去するとともに，スパムテンプレートとして保存する

4. (1) に戻る

この操作を N 回繰り返し，N 種類のスパムテンプレートを検出した．このようにして得られたスパムテンプレートを含むブログエントリを splog と判定した．検出するスパムテンプレートの数とフィルタリングの性能の関係を表 4.10 に示す．表に示されているように，スパムテンプレート数の増加にともなって，フィルタリング性能も向上するが，テンプレート数 170 くらいで，性能の改善がほとんど得られなくなる．これ以上，検

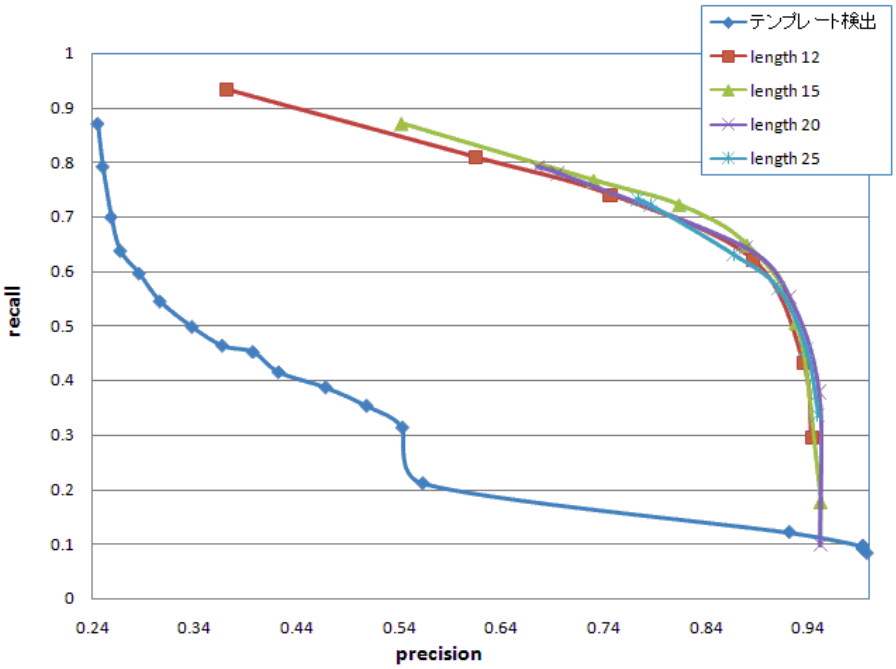

図 4.15: スパムテンプレート検出との比較

出回数を増やすと「よろしくお願いします」「ありがとうございました」など，スパムテンプレートではない定型的な語句が出てくるようになり，フィルタリング性能の劣化が起こるためである．

提案手法では，図 4.9 や図 4.10 に示すように最大で 0.75 程度の F 値が得られており，本論文で行った実験の範囲では，提案手法のほうが splog フィルタリングに適していた．

スパムテンプレートに基づく splog フィルタリングは，表 4.10 に示されているように，少数のテンプレートに関しては高い presicion を実現できる．このことから，顕著なスパムテンプレートは正確に発見できていることが予想される．一方で，全体的に recall は非常に低く，splog を広く検出することができない．

十分な大きさで同一テンプレートをもった splog がコーパス中になければならないため，コーパスに表れない多種多様なテンプレートの検出が難しい．つまり，出現頻度の小さいテンプレートの発見が出来ないため，高い F 値を実現できないと考えられる．図 4.15 にこの結果と，提案手法 (Unlabel entries + Search API) 表 4.7 の precision-recall カーブを示す．

なお，成澤ら [51] の研究は，スパムテンプレートを検出することを主たる目的としており，本実験で splog フィルタリングへ適用するあたって，チューニング等は行わなかった．次章と次節で述べるように，splog のタイプと手法との適性についてさらに分析をすすめる必要がある．

4.8 splog のタイプ別評価

splog に様々な種類があることがテストデータ作成後にわかってきた．テストデータからランダムサンプリングを行い，の構成比を調べた．その結果を図 4.16 および 表 4.11 に示す．

search results，product induction，template decorator の 3 種類及びそれらの複合型が splog の大部分を占めている．これらはそれぞれ，その目的に応じた使われ方がされていると推測される．

search results の場合は，自身が検索されることを目的とし，クリック広告による収入と，他サイトの検索順に向上を狙っている．product induction，template decorator は，直接的に商品の宣伝と販売を狙っていると思われる．そして，現状，これらを「生産」する splog ツールの割合が多いということが予想される．

本稿では，提案手法とテンプレート検出法について，各 splog 種別に対する適正を調べた．

作成した各コーパス，Labeled entries，Unlabel entries，Search API，Unlabel entries+Search API への splog 種別評価として，表 4.17-4.20 がそれぞれ対応する．これらの値は，表 4.4，図 4.16 でサンプリングしたそれぞれの splog に対する recall の値である．

同じく，テンプレート検出法 による結果を表 4.12-4.16 に示す．こちらも同様に 表 4.4，図 4.16 でサンプリングした splog に対する recall の値である．

提案手法，テンプレート検出法のどちらも，splog 種別に顕著な検出精度の差はなく，同程度の値である．特にテンプレート検出法の方は，template decorator に分類される splog を検出することを目的としているために，template decorator に対する性能は他に比べて高くなることが期待されたが，必ずしもそうはならなかった．逆に，テンプレート検出法では検出が困難であるはずの search results なども捉えられる場合がある．

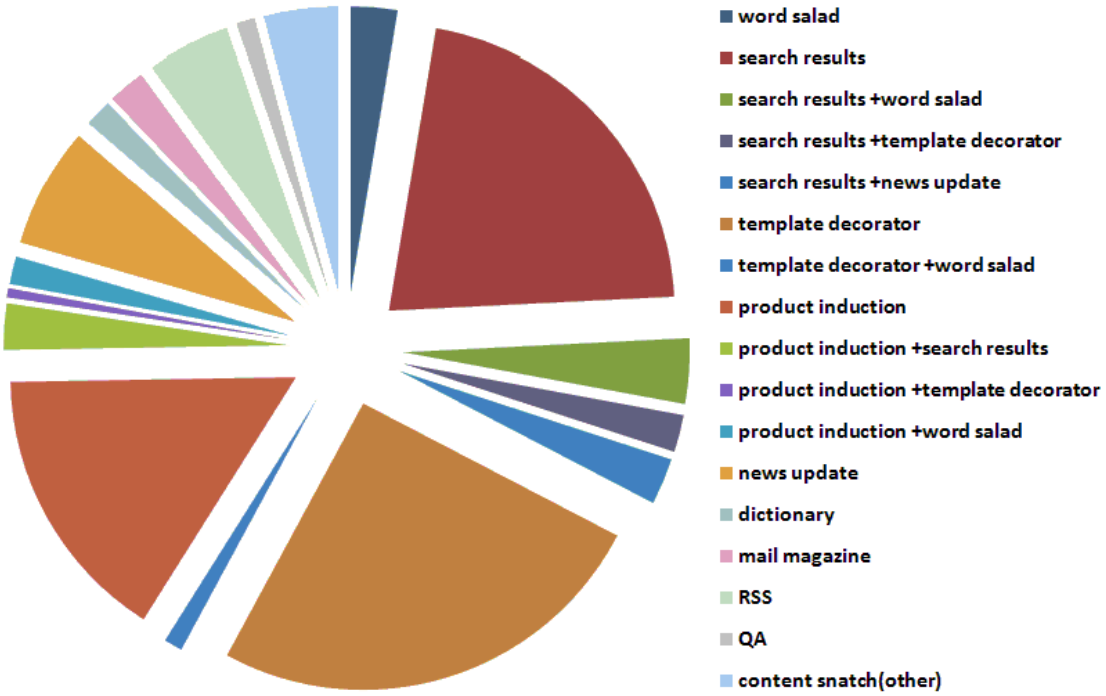

図 4.16: splog 構成比

コンテンツ中の文字列の一致から splog を検出しようとする手法では，概してこのような結果になるという可能性を示している．

各種別，news update，product induction，template decorator について平均以下である理由であるが，コピー文字列が検出できず，オリジナルコンテンツであると誤認しているため検出できないと推測される．news update の場合は同一のニュース記事を他に見つけられなかった場合，product induction は同一の商品，template decorator は同一の template である．

search results が予想よりも高い精度で検出できている理由はこの逆である．検索結果コンテンツ中に，同一の（部分）コンテンツが複数回，参照されているためである．このコピーされている部分は，ブログエントリ中では一部分にしかならないが，提案手法ではこの”部分”が存在することこそが，splog であると判断する根拠になったためと思われる．

4.9 提案手法の有効性を発揮するためのシステム設計

本手法は，当初は一般的な splog のフィルタリングを行うための手法として研究をしてきたが，その性能と，特に時間的効率で不十分な面が残った．しかしながらより有効な splog filter を構築するために，様々な試行を行う過程で，提案手法の有効な利用方法がわかってきた．

本手法は予備知識を使わずに未知の splog を検出するという強力な特徴がある．その splog を検出する際に特定されたコピー文字列 *template* に注目すると，*template* が含まれるブログエントリが splog である可能性が非常に高いという場合がある．このようにして「不自然」な文字列を選択し，そのコピー文字列を含むブログエントリを splog と判定する．

そこで提案手法を，既知の splog をパターンからフィルタリングする splog filter と組み合わせ，splog filter のチェックを通り抜けた未知パターンを持つ splog を検出し，検出した splog を手動でチェックし，新しいパターンを splog filter に反映させることによって，より効果的な splog filter を構築することができると思われる．

図 4.17， 図 4.18 に，このような splog filter の構成を示す．

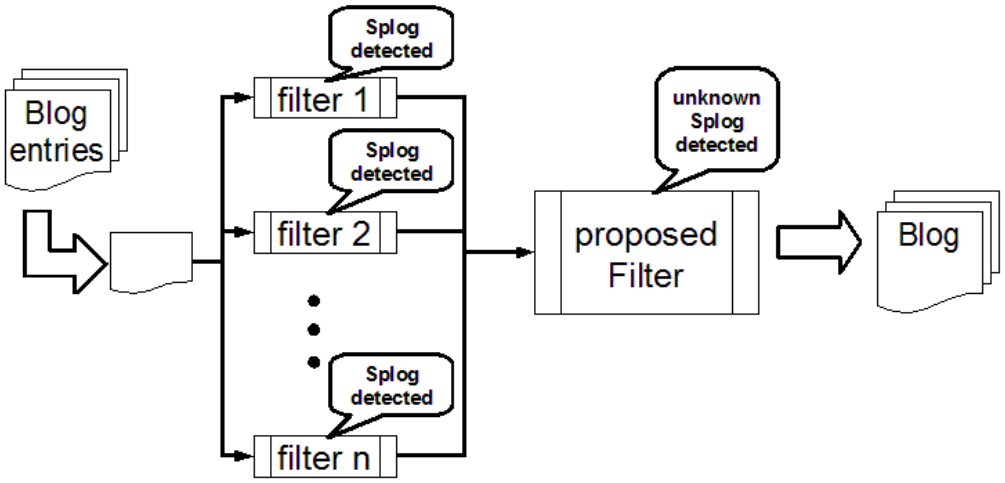

図 4.17: 有効なシステム設計

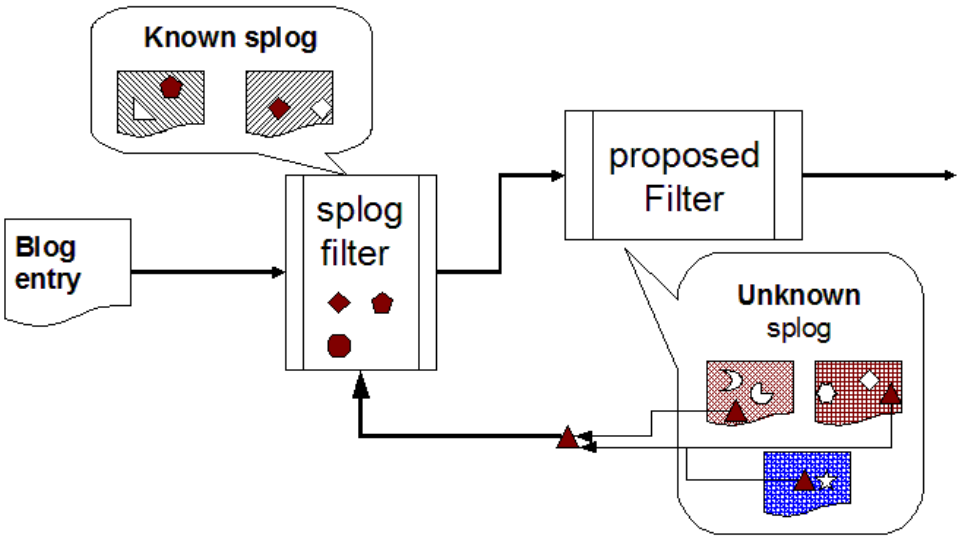

図 4.18: splog フィードバック

l = 12			
size (Mbyte)	precision	recall	F
26.60	0.749	0.682	0.714
42.97	0.722	0.734	0.728
58.02	0.733	0.730	0.731
62.16	0.732	0.740	0.736
151.97	0.746	0.741	0.743
l = 15			
size (Mbyte)	precision	recall	F
26.60	0.752	0.648	0.696
42.97	0.725	0.728	0.726
58.02	0.743	0.723	0.733
62.16	0.736	0.746	0.741
151.97	0.757	0.752	0.754
l = 20			
size (Mbyte)	precision	recall	F
26.60	0.735	0.648	0.689
42.97	0.728	0.698	0.713
58.02	0.731	0.705	0.717
62.16	0.725	0.731	0.728
151.97	0.792	0.712	0.750
l = 25			
size (Mbyte)	precision	recall	F
26.60	0.817	0.559	0.664
42.97	0.882	0.567	0.690
58.02	0.809	0.640	0.715
62.16	0.809	0.658	0.726
151.97	0.773	0.732	0.752

表 4.8: データベースサイズとフィルタリング性能

length	実行時間/ブログエントリ (msec)
30	410.68
25	422.09
20	443.06
15	458.55
12	490.12
10	487.34
7	533.18
5	566.32
4	603.28
3	661.99
2	800.31
1	1073.01

表 4.9: 最小コピー文字列長と処理時間

Template 数	precision	recall	F
10	1.000	0.004	0.008
30	1.000	0.009	0.018
50	1.000	0.018	0.035
70	1.000	0.038	0.072
90	1.000	0.078	0.144
100	0.997	0.082	0.152
110	0.993	0.091	0.166
120	0.993	0.091	0.166
130	0.993	0.096	0.175
140	0.921	0.121	0.214
150	0.563	0.211	0.306
160	0.543	0.313	0.398
170	0.508	0.353	0.417
180	0.468	0.387	0.424
190	0.422	0.415	0.418
200	0.397	0.452	0.423
220	0.366	0.463	0.409
240	0.337	0.498	0.402
260	0.306	0.545	0.392
280	0.285	0.596	0.386
300	0.267	0.637	0.376
320	0.258	0.700	0.377
350	0.250	0.792	0.380
400	0.244	0.871	0.382

表 4.10: スパムテンプレート検出によるフィルタリング

splog 種別	検出数
word salad	5
search results	41
search results +word salad	7
search results +template decorator	4
search results +news update	5
template decorator	48
template decorator +word salad	2
product induction	30
product induction +search results	5
product induction +template decorator	1
product induction +word salad	3
news update	13
dictionary	3
mail magazine	4
RSS	9
QA	2
content snatch(other)	8

表 4.11: splog 構成比

Template 数	10	30	50	70	90
content snatch	0.000	0.000	0.000	0.000	0.000
dictionary	0.000	0.000	0.000	0.000	0.000
mail magazine	0.000	0.000	0.500	0.500	0.500
search results +news update	0.000	0.200	0.200	0.400	0.400
news update	0.000	0.000	0.000	0.000	0.000
product induction	0.000	0.000	0.000	0.000	0.000
search results +product induction	0.000	0.000	0.000	0.000	0.571
product induction +template decorator	0.000	0.000	0.000	0.000	0.000
product induction +word salad	0.000	0.000	0.000	0.000	0.000
qa	0.000	0.000	0.000	0.000	0.000
rss	0.000	0.000	0.000	0.111	0.111
search results +word salad	0.000	0.000	0.000	0.000	0.200
search results	0.000	0.000	0.000	0.024	0.073
template decorator +search results	0.000	0.000	0.000	0.000	0.000
template decorator +word salad	0.000	0.000	0.000	0.000	0.000
template decorator	0.021	0.063	0.104	0.146	0.146
word salad	0.000	0.000	0.000	0.000	0.000

表 4.12: スパムテンプレート検出による splog 種別性能 1

Template 数	100	120	130	140	150
content snatch	0.000	0.000	0.143	0.286	0.286
dictionary	0.000	0.000	0.667	0.667	0.667
mail magazine	0.500	0.500	0.500	0.500	0.500
search results +news update	0.400	0.400	0.400	0.600	0.800
news update	0.000	0.000	0.000	0.154	0.308
product induction	0.000	0.000	0.000	0.000	0.067
search results +product induction	0.571	0.571	0.571	0.571	0.714
product induction +template decorator	0.000	1.000	1.000	1.000	1.000
product induction +word salad	0.000	0.000	0.000	0.000	0.333
qa	0.000	0.000	0.000	0.000	0.000
rss	0.111	0.111	0.111	0.111	0.333
search results +word salad	0.200	0.200	0.200	0.200	0.400
search results	0.073	0.098	0.098	0.098	0.268
template decorator +search results	0.000	0.000	0.000	0.000	0.250
template decorator +word salad	0.000	0.000	0.000	0.000	0.000
template decorator	0.146	0.146	0.146	0.167	0.250
word salad	0.000	0.000	0.000	0.000	0.000

表 4.13: スパムテンプレート検出による splog 種別性能 2

Template 数	160	170	180	190	200
content snatch	0.286	0.286	0.286	0.286	0.286
dictionary	0.667	0.667	0.667	0.667	0.667
mail magazine	0.500	0.500	0.500	0.500	0.500
search results +news update	0.800	0.800	0.800	0.800	0.800
news update	0.308	0.308	0.308	0.385	0.462
product induction	0.133	0.133	0.167	0.167	0.233
search results +product induction	0.714	0.714	0.714	0.714	0.714
product induction +template decorator	1.000	1.000	1.000	1.000	1.000
product induction +word salad	0.333	0.333	0.333	0.333	0.333
qa	0.000	0.000	0.500	0.500	0.500
rss	0.444	0.444	0.556	0.556	0.556
search results +word salad	0.400	0.400	0.400	0.400	0.400
search results	0.390	0.439	0.512	0.537	0.537
template decorator +search results	0.500	0.500	0.500	0.500	0.500
template decorator +word salad	0.000	0.000	0.000	0.000	0.000
template decorator	0.417	0.438	0.500	0.521	0.563
word salad	0.000	0.200	0.200	0.200	0.200

表 4.14: スパムテンプレート検出による splog 種別性能 3

Template 数	220	240	260	280	300
content snatch	0.286	0.286	0.286	0.571	0.571
dictionary	0.667	0.667	0.667	0.667	0.667
mail magazine	0.500	0.500	0.500	0.500	0.750
search results +news update	0.800	0.800	0.800	0.800	0.800
news update	0.462	0.538	0.692	0.692	0.692
product induction	0.233	0.233	0.267	0.433	0.433
search results +product induction	0.714	0.714	0.714	0.714	0.857
product induction +template decorator	1.000	1.000	1.000	1.000	1.000
product induction +word salad	0.333	0.333	0.667	0.667	0.667
qa	0.500	0.500	0.500	0.500	0.500
rss	0.556	0.556	0.667	0.778	0.889
search results +word salad	0.400	0.400	0.400	0.800	0.800
search results	0.537	0.610	0.732	0.756	0.780
template decorator +search results	0.500	0.500	0.750	0.750	0.750
template decorator +word salad	0.000	0.000	0.000	0.000	0.000
template decorator	0.563	0.604	0.688	0.708	0.729
word salad	0.200	0.200	0.200	0.400	0.400

表 4.15: スパムテンプレート検出による splog 種別性能 4

Template 数	320	350	400
content snatch	0.571	0.714	0.714
dictionary	1.000	1.000	1.000
mail magazine	1.000	1.000	1.000
search results +news update	0.800	0.800	0.800
news update	0.769	0.846	0.846
product induction	0.500	0.767	0.867
search results +product induction	1.000	1.000	1.000
product induction +template decorator	1.000	1.000	1.000
product induction +word salad	0.667	0.667	0.667
qa	0.500	0.500	1.000
rss	0.889	0.889	0.889
search results +word salad	1.000	1.000	1.000
search results	0.902	0.951	0.976
template decorator +search results	0.750	0.750	0.750
template decorator +word salad	0.000	0.500	1.000
template decorator	0.750	0.792	0.833
word salad	0.600	0.800	0.800

表 4.16: スパムテンプレート検出による splog 種別性能 5

Labeled entries				
l	20	15	12	10
content snatch	0.714	0.714	0.714	0.714
dictionary	1.000	1.000	1.000	1.000
mail magazine	1.000	1.000	1.000	1.000
search results +news update	1.000	1.000	1.000	1.000
news update	0.615	0.538	0.615	0.692
product induction	0.566	0.566	0.600	0.600
search results +product induction	1.000	1.000	1.000	1.000
product induction +template decorator	1.000	1.000	1.000	1.000
product induction +word salad	0.666	0.666	0.333	0.666
qa	0.000	0.000	0.000	0.500
rss	0.888	1.000	1.000	1.000
search results +word salad	0.800	1.000	1.000	1.000
search results	0.730	0.707	0.756	0.780
template decorator +search results	1.000	0.75	1.000	1.000
template decorator +word salad	1.000	1.000	1.000	1.000
template decorator	0.645	0.687	0.604	0.625
word salad	0.400	0.400	0.800	1.000

表 4.17: Labeled entries による splog 種別性能

Unlabel entries				
l	20	15	12	10
content snatch	0.714	0.714	0.571	0.571
dictionary	1.000	1.000	1.000	0.666
mail magazine	1.000	1.000	1.000	1.000
search results +news update	1.000	1.000	1.000	1.000
news update	0.538	0.538	0.615	0.615
product induction	0.666	0.733	0.733	0.666
search results +product induction	1.000	1.000	1.000	1.000
product induction +template decorator	1.000	1.000	1.000	1.000
product induction +word salad	0.666	0.666	0.333	0.666
qa	0.000	0.000	0.500	0.500
rss	1.000	1.000	1.000	1.000
search results +word salad	1.000	1.000	1.000	1.000
search results	0.804	0.804	0.804	0.804
template decorator +search results	1.000	0.750	1.000	1.000
template decorator +word salad	1.000	1.000	1.000	1.000
template decorator	0.708	0.750	0.687	0.687
word salad	0.400	0.400	0.800	0.800

表 4.18: Unlabel entries による splog 種別性能

Search API				
l	20	15	12	10
content snatch	0.714	0.714	0.714	0.714
dictionary	1.000	1.000	1.000	0.666
mail magazine	1.000	1.000	1.000	1.000
search results +news update	1.000	1.000	1.000	1.000
news update	0.692	0.692	0.692	0.692
product induction	0.633	0.666	0.533	0.600
search results +product induction	1.000	1.000	1.000	1.000
product induction +template decorator	1.000	1.000	1.000	1.000
product induction +word salad	0.666	0.666	0.333	0.666
qa	0.000	0.000	0.000	0.500
rss	1.000	1.000	1.000	1.000
search results +word salad	1.000	1.000	1.000	1.000
search results	0.878	0.878	0.902	0.926
template decorator +search results	1.000	0.750	1.000	1.000
template decorator +word salad	1.000	1.000	1.000	1.000
template decorator	0.645	0.687	0.625	0.625
word salad	0.400	0.600	0.800	0.800

表 4.19: Search API による splog 種別性能

Unlabel entries+Search API				
l	20	15	12	10
content snatch	0.714	0.714	0.857	0.571
dictionary	1.000	1.000	0.666	0.666
mail magazine	1.000	1.000	1.000	1.000
search results +news update	1.000	1.000	1.000	1.000
news update	0.692	0.692	0.769	0.692
product induction	0.566	0.633	0.633	0.600
search results +product induction	1.000	1.000	1.000	1.000
product induction +template decorator	1.000	1.000	1.000	1.000
product induction +word salad	0.666	0.333	0.333	0.666
qa	0.000	0.500	0.500	0.500
rss	1.000	1.000	1.000	1.000
search results +word salad	1.000	1.000	1.000	1.000
search results	0.878	0.902	0.902	0.878
template decorator +search results	1.000	1.000	0.750	0.750
template decorator +word salad	1.000	1.000	1.000	1.000
template decorator	0.708	0.729	0.708	0.708
word salad	0.600	0.800	0.800	0.800

表 4.20: Unlabel entries+Search API による splog 種別性能

第5章　局所テキストアライメントに基づいた複数文書要約

5.1　要約手法

5.1.1　要約手法の概要

典型的な複数文書要約は，記述されている全ての情報から，同一情報の重複を省き，それぞれユニークな情報をただ一つずつ並べることが行われる．主に対象とするデータはニュースサイトの記事であるが，これら単一のイベントに関する記事では同じ語句や同じ言い回しが頻出するという性質がある．

本論文で提案する手法は，コーパス中の（2箇所以上の）任意の箇所に出現する任意の類似表現を充分に小さい計算量で検出し，重複を省き (図 5.1) 1度づつ列挙する手法である．

複数の文書で繰り返し現れる情報は重要であるとした仮定に基づいている．類似の定義としては，文字列（記号列）二つの組の間で計算できるものであれば，どのようなものでも提案手法のアルゴリズムには組み込むことができる．今回は文字列を形態素解析によって単語に分割し，単語レベルでの重み付編集距離を採用した

5.1.2　要約に用いる候補表現の抽出

本稿で提案する手法はクラスタ生成に関するものであり，ここを中心に議論し，これによる冗長性削減が要約に貢献できることを示す．提案手法で抽出する文のクラスタは，カテゴリにおいて同一性 [56] に基づく関係に分類される文の組である．クラスタリングの計算コストが大きい

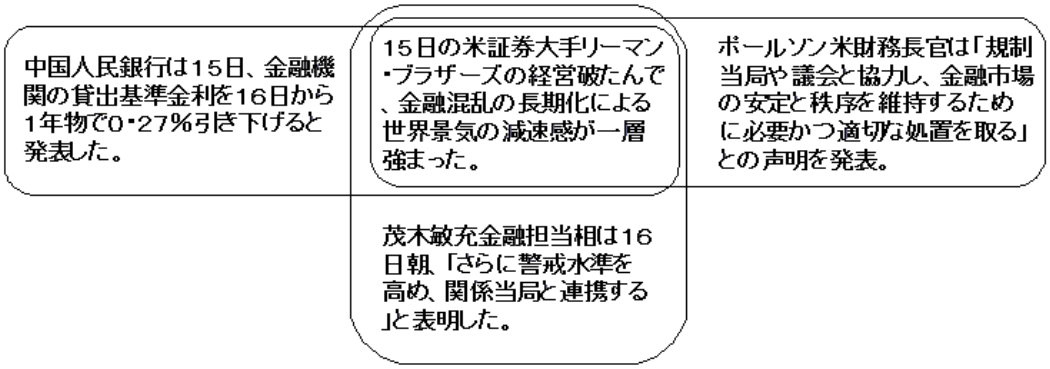

図 5.1: 要約文生成イメージ

ため，本研究ではこの計算量を抑える類似表現クラスタリング手法を考案した．

部分単語列生成

まず以下に示すように全ての可能な部分単語列集合 S を取得する．あるトピックが文書 $\{D_1, D_2, \cdots, D_n\}$ で構成されているとし，文書 D_i は 単語列（文）$\{W_{i1}, W_{i2}, \cdots, W_{im_i}\}$ で構成され，単語列 W_{ij} は語 $w_{ij1}w_{ij2}\cdots w_{ijl_{ij}}$ で構成されるとする．すると部分単語列集合 S は以下の式で表わされる．

$$S \equiv \bigcup_{i,j} \bigcup_{1 \le k \le l \le |W_{ij}|} \{w_{ijk} \cdots w_{ijl}\} \tag{5.1}$$

ここで，$|W_{ij}|$ は，単語列 W_{ij} に含まれる単語数を表す．

例えば，「12 日に辞任表明を行った安倍晋三首相 (52) は，13 日に慶応大学病院に入院した．」といった文からは，「12 日」「12 日に辞任表明」といったプリフィックや「辞任表明を行った」「辞任表明を行った安部晋三首相」など，単語列（文）中に含まれる任意の部分単語列が S に含まれる．図 5.2 に示すように，全ての始点と終点の組み合わせを列挙する．

索引生成

クラスタリングを効率的に行うため，S 中の単語列の索引を作成する．集合 S に含まれる任意の部分単語列 W に対して，W に含まれる単語を

IDF(Inverse Document Frequency)

$$idf(w) \equiv log\frac{|D|}{df(w)} \tag{5.2}$$

で降順に整列した単語列

$$idx(W) \equiv w'_1 w'_2 ... w'_l \tag{5.3}$$

を考える．つまり，$i < j$ のとき $idf(w'_i) \geq idf(w'_j)$ が成り立つ．この並び替えた単語列 $idx(W)$ を，W の索引と呼ぶ．式 (5.2) において，$|D|$ はコーパス全ての文書の数を，また $df(w)$ は w が出現する文書の数を表している．例えば，部分単語列「安部首相は慶応大病院」において，各単語の IDF 値が表 5.1 で与えられるとする．すると,「安部首相は慶応大病院」の索引は以下のようになる．

「慶応 安部 首相 病院 大 は」

提案手法では，上記の索引を辞書順 [20] に並び替えて用いる．(図 5.3) 本稿では，単語の IDF 値を，その単語を含む部分単語列におけるその単語の意味的貢献の度合とみなし，索引を辞書順に並べることによって，類似する部分単語列が索引上の近い位置に配置されることをねらっている．

5.1.3 配列整理

部分単語列 W_1 と W_2 の編集距離が小さい場合，W_1 と W_2 のプリフィックス同士の編集距離も小さくなる．そこで，S 中で，索引のリスト中で隣接する部分単語列を削除する．このようにして得られた単語列の索引リストを I とする．

例えば，S 中で部分単語列が以下の順番で並んでいたとする．

1. 政策実行力不足を理由に突然の辞任を
2. 政策実行力不足を理由に突然の辞任を表明していた
3. 突然の辞任を表明していた

(1)，(3) は共に (2) の部分単語列である．(1)，(2)，(3) のどれもが，同一の文書の中の部分文字列であれば (1)，(3) を削除する．このようにし

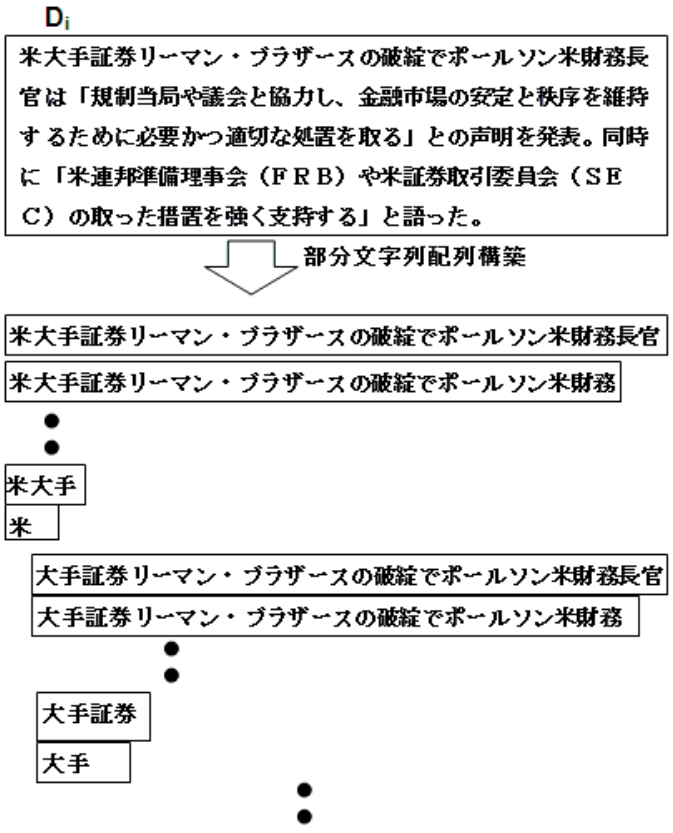
Di
米大手証券リーマン・ブラザースの破綻でポールソン米財務長官は「規制当局や議会と協力し、金融市場の安定と秩序を維持するために必要かつ適切な処置を取る」との声明を発表。同時に「米連邦準備理事会（ＦＲＢ）や米証券取引委員会（ＳＥＣ）の取った措置を強く支持する」と語った。
部分文字列配列構築
米大手証券リーマン・ブラザースの破綻でポールソン米財務長官
米大手証券リーマン・ブラザースの破綻でポールソン米財務
米大手
米
大手証券リーマン・ブラザースの破綻でポールソン米財務長官
大手証券リーマン・ブラザースの破綻でポールソン米財務
大手証券
大手

図 5.2: 部分単語列列挙

て，隣接する部分単語列同士の一方がもう一方の部分にならないように索引リストを整理する．この処理の直感的な意味は，極大な部分単語列によって，包含する部分単語列を代替させることである．

I 中の i 番目の単語列 $idx(W)$ を $I(i)$ とする．$I(a) \subset I(b)$ とは， $I(a)$ が $I(b)$ の部分文字列であることを表す．$Docof(I(a))$ とは， I(a) の元の単語列があった文書の番号を示す．

このとき， I は以下の条件を満たす．

> $Docof(I(i)) = Docof(I(i+1))$ である時， $I(i) \subset I(i+1)$ と $I(i) \supset I(i+1)$ のどちらも成り立たない

索引リスト I は，元の部分単語列配列への順列付け用ポインタであり，後述するクラスタリングの際に実際に使用する部分単語列は，単語列 W であることに注意されたい．

5.1.4 候補表現のクラスタリング

単語列類似度の定義

本研究では 文字単位ではなく単語（形態素）単位での重み付編集距離 [15] によって単語列間の類似度を計算する．コストを出現頻度から計算することに本手法の特徴がある．

編集コストは (5.2) 式で与えられる IDF 値 によって決定される．文中の語 w の挿入コスト c_i ， 削除コスト c_d は以下の値を用いる．

$$c_i(w) = c_d(w) \equiv idf(w)$$

低頻度語には大きい編集コストを，ストップワードのような頻出語には小さい編集コストを割り当てる．語 w_1 を 語 w_2 に変換する編集コストを以下のように定義する．

$$c_s(w_1, w_2) \equiv \begin{cases} c_{max} + \dfrac{c_{min}^2}{c_{max}} & w_1 \neq w_2 \\ -2C_{max} & w_1 = w_2 \end{cases} \tag{5.4}$$

ここで

$$\begin{aligned} c_{min} &\equiv min(idf(w_1), idf(w_2)) \\ c_{max} &\equiv max(idf(w_1), idf(w_2)) \end{aligned} \tag{5.5}$$

これは，ほぼ同等コストの単語の置換には，二つの単語の合計に近いコストを，コストに大きな開きがある場合，大きいコストの値に近い値を用いることを意味する．$w_1 = w_2$ である場合のコストが負の値になる．テキスト処理においては，一般的に，同一語の置換コストは 0 となる．一方，バイオインフォマティクスにおけるアラインメントでは，このように同一語の置換コストに得点を与える事が一般的であり，本手法でも得点を与えている．この効果は，IDF が大きい重要な語の一致には大きなスコアを与えるためである．例えば，以下の文を考える．

1. 与謝野氏によると，首相の健康管理を担当する医師が「疲労がピークに達しており...

2. 与謝野官房長官は同日午前の記者会見で，首相が体調不良のため病院で検査を受けた結果,「医師が『疲労がピークに達しており...

提案手法では，先頭の「与謝野」には大きなコストが割り当てられている．この得点によって「疲労がピークに達しており」までの後続のコストが小さい普通名詞の編集コストをカバーする．このような方法で，特に助動詞などの語尾，助詞の変化，時制の変化，敬称，普通名詞の編集をカバーする狙いがある．$w_1 \neq w_2$ であるときのコストは，語のスコアに大きな開きがある場合に そのスコアの比に応じて小さいスコアの影響が小さくなるように定義されている．スコアが近い場合は，スコアの合計に近い値になり，スコアに差がある場合は大きいほうのスコアに近づく．

- $w_1 = 1, w_2 = 1$ である場合，
  $c_s(w_1, w_2) = 1 + \frac{1^1}{1} = 2$.

- $w_1 = 1, w_2 = 2$である場合，
  $c_s(w_1, w_2) = 2 + \frac{1^2}{2} = 2.5$.

- $w_1 = 1, w_2 = 10$ の時は，
  $c_s(w_1, w_2) = 10 + \frac{1^2}{10} = 10.1$.

これは置換コストが挿入/削除コストよりも小さいということを近似的に表現するための式である．提案手法では，同義語辞書などは使用しないため同義語の判定ができないが，上記の編集コストを導入することによって，同義語による言い替えのコストを下げる．例えば，以下の文を考える．

安倍	首相	総理	は	13 日	慶応	大	病院
2.81	2.62	2.80	0.42	1.17	3.88	1.36	1.84

表 5.1: IDF 値の例

	安倍	首相	は	13 日	慶応	大	病院
安倍	0	2.62	3.04	4.21	8.09	9.45	11.29
総理	2.80	5.25	5.48	6.33	10.11	11.55	13.39
は	3.22	5.49	4.41	5.58	9.46	10.82	12.66
13 日	4.39	6.36	5.58	2.07	5.95	7.31	9.15
慶応	8.27	10.04	9.46	5.95	0	1.36	3.2
病院	10.11	12.18	11.3	8.53	1.84	2.85	0

表 5.2: 局所アラインメントの計算例

1. 安倍首相が機能性胃腸炎で入院し...

2. 安倍首相は，潰瘍性大腸炎で入院へ...

上記のような珍しい固有名詞と，普通名詞に近い一般的な語句との言い換えのコストを十分に減らすことを期待して式を設定している．

各語の編集コストを用いて DP マッチングによって部分単語列の類似度を計算する．単語列 $W = w_1 w_2 \cdots w_m$ について，W_j は W の長さ j のプリフィックスを表すものとする．単語列 W と 単語列 Z について，それらのプリフィックス同士の距離を次のように再帰的に定義する．

$$
\begin{aligned}
& C(W_i, Z_j) \\
& \quad \equiv \quad \min \begin{cases} max(C(W_{i-1}, Z_{j-1}) + c_s(w_i, z_j), 0) \\ C(W_{i-1}, Z_j) + c_d(w_i) \\ C(W_i, Z_{j-1}) + c_i(z_j) \end{cases}
\end{aligned} \tag{5.6}
$$

例として， IDF 値が 表5.1 で与えられたとき,「安倍首相は 13 日 慶応大病院」 と 「安倍総理は 13 日慶応病院」 との，最適経路を求めるためのコスト行列を表 5.2 に示す.

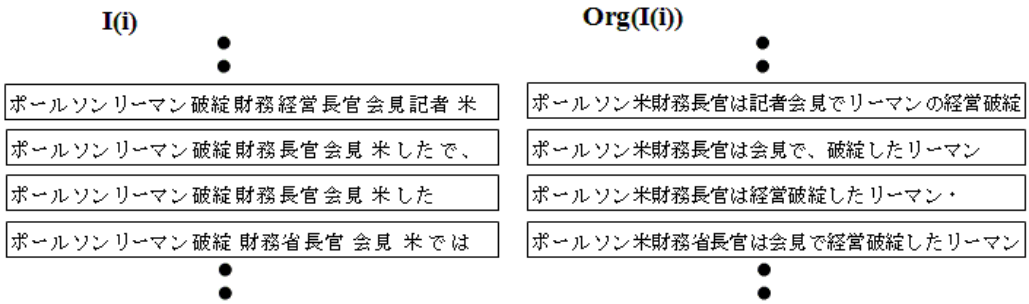

図 5.3: 単語列ソート

単語列とクラスタ間の一致長定義

本稿で述べるシステムにおいてクラスタリングがもっとも計算コストが高い．そこで，索引リスト I を用いて効率の良いクラスタリングを行う．まず，任意のプリフィックスの組 W_i, Z_j とパラメタ τ に対して，類似接頭単語列長を以下のように定義する

$$l(W_i, Z_j, \tau) \equiv \begin{cases} max(|W_i|, |Z_j|) & C(W_i, Z_j) < \tau \\ 0 & otherwise \end{cases} \tag{5.7}$$

そのとき，単語列 W とクラスタ T の類似一致長を以下のように定義する．

$$match(W, T, \tau) \equiv \max_{Z \in T, i, j > 0} l(W_i, Z_j, \tau) \tag{5.8}$$

これはクラスタ T 中の W との距離が閾値 τ を超えない最大の一致領域長 を意味する．e を，類似一致長の下限閾値とし，全てのクラスタの中で $match(W, T, \tau) \geq e$ の最大値を示す単語列 が T 中にあれば， W を T に割り当てる．

$match(W, T, \tau) < e$ である場合は，W を新しいクラスタとする．この一連の計算のとき，式 (5.8) を用いるが，もし全ての単語列の距離を計算しようとすると多大な計算量が必要となる．このため，上記の整列された索引において，互いに距離の短い単語列は十分に近い領域に集まっていると考え近似的なクラスタリングを行う．

1. $T \leftarrow \{\}, Cand \leftarrow \{\}$
2. for $i = 1$ to $|I|$
3. $\quad W \leftarrow org(I(i))$
4. $\quad C_m \leftarrow arg\ max_{C \in Cand}\, match(W, C, \tau)$
5. $\quad$ if $match(W, C_m, \tau) > e$ then
   $\quad\quad C_m \leftarrow C_m \cup \{W\}$
   $\quad$ else
   $\quad\quad Cand \leftarrow Cand \cup \{W\}$
6. $\quad$ if $|Cand| > \lambda$ then
   $\quad\quad C_f \leftarrow arg\ min_{C \in Cand}\ match(W, C, \tau)$
   $\quad\quad Cand \leftarrow Cand - C_f$
   $\quad\quad$ if $|C_f| > 1$ then
   $\quad\quad\quad T \leftarrow T \cup \{C_f\}$
7. $T \leftarrow T \cup \{Cand\}$
8. T を返す

図 5.4: クラスタリングアルゴリズム

クラスタリングアルゴリズム

以上を踏まえて，クラスタリングアルゴリズムを図5.4に示す．アルゴリズムへの入力として整列された索引リスト I を与える．出力は単語列クラスタ T である．また，I 中の i 番目の単語列 $idx(W)$ の元の単語列 W を $org(I(i))$ とする．

このアルゴリズムで $Cand$ はクラスタ候補を保持するために用いられる．λ は $Cand$ が保持するクラスタ候補数の上限値を指定するパラメータである．I から W を順に一つずつ取り出し，類似一致長が最大かつその値が閾値 e 以上のクラスタを選択する．

提案手法では，この計算を $O(\lambda|I|)$ 回行うことになる．通常，$\lambda \ll |I|$ であり，要素数の線形に比例する計算量ですむ．一方，階層的クラスタリングでは，$O(|I|^2)$ 回の類似度計算が必要であり，$|I|$ の大きさから，こ

れは効率的ではない．k-means 法 (k:クラスタ数) の場合， $O(|I|k)$ 回の類似度計算を収束するまで繰り返す必要がある．提案手法で必要となる部分単語列のクラスタリングでは，多くの部分単語列が単独クラスタとなるため，クラスタ数 $k \approx |I|$ となる．このため， k-means 法の計算量は $O(|I|^2)$ に近い値になり，現実的ではない．このような理由から，提案手法では，既存のクラスタリング手法を用いず，図 5.4 に示すアルゴリズムを用いている．

k-means 法は事前にクラスタ数を指定しなければならないが，提案手法の場合は，アルゴリズムの終了時点でクラスタ数を決定できる．これも提案手法の利点である．

このクラスタリング過程は充分に類似している要素集合だけを残すことが目的であり，その意味で既存のクラスタリングの問題設定とは異なる問題を扱っている．

5.1.5 クラスタからの要約生成

提案手法では，クラスタリング (図 5.7) によって得られたクラスタ集合から要約文を生成する．各クラスタについて，そのクラスタを代表する文を選択し，列挙することで要約文を生成する．(図 5.5)

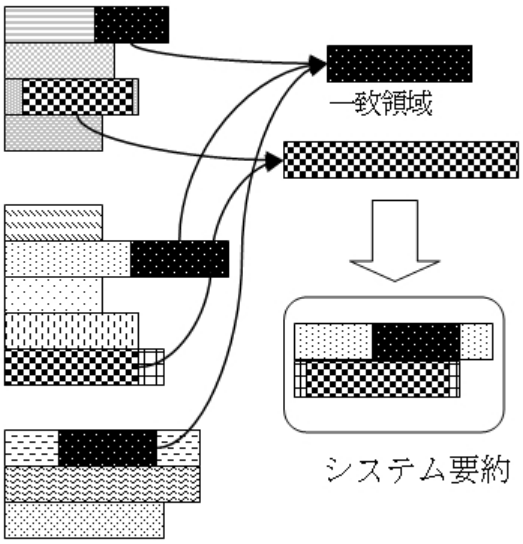

図 5.5: 抜粋要約生成

入力: 単語列クラスタ: T, 要約長: $Limit$
出力: 要約文 $Summary$

1. T を，クラスタの大きさの降順に整列
2. $Summary \leftarrow \{\}$
3. **while** $|Summary| < Limit$
4. 　i 番目のクラスタ C_i の代表文 $S(C_i)$ を取得する
5. 　$Summary$ に $S(C_i)$ を追加
6. **return** $Summary$ を返す.

図 5.6: 要約文生成手続き

あるクラスタ C を代表する文として，各語の IDF 値の合計がクラスタ C 中で最も大きい文を用いる.

$$S(C) = arg\,\max_{s}(\sum_{w \in s} idf(w)) \tag{5.9}$$

本稿では，複数文書要約を行うこの提案手法を TA(Text-Alignment) と名付ける.

一般的に要約生成で選択する文は，同じ情報を持っている場合より短い方が望ましい．クラスタを代表する文を，大きいクラスタから順に抽出していき，要約文の制限を越えない長さまで続ける．この処理を図 5.6 に示す.

5.2 実験

5.2.1 ベンチマークデータ

提案手法の重複情報削減性能を定量的に評価するため NTCIR4 [11] の複数文書要約タスク TSC3 のコーパスを使用した．コーパスは毎日新聞

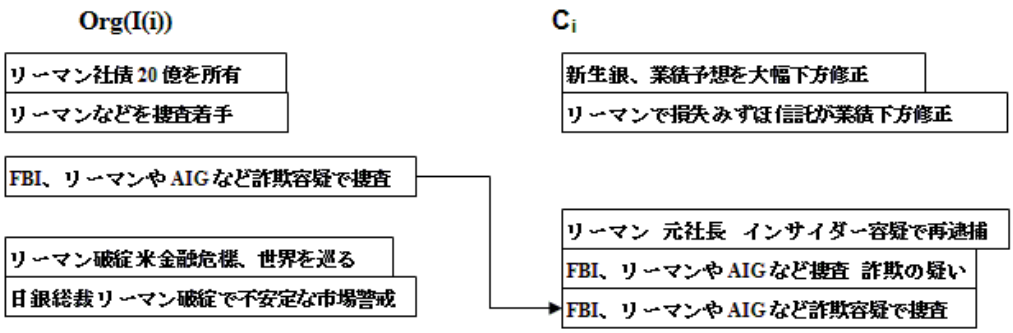

図 5.7: クラスタリング過程

と読売新聞の 1998 年から 1999 年の記事から構成される．30 のニュースクラスタが与えられており，そのほとんどはシングルイベント [21] のニューストピックに相当する．

コーパスでは予め重要文集合 $\{m_1, m_2, ..., m_n\}$ が各トピックに与えられており，それぞれ正解要約 (long) と (short) の 2 種類が存在する．この重要文はトピックの要点となる情報を持っており，これらが要約文となるのであるが，その組み合わせは一通りではない．

i 番目の重要文 m_i に等しい情報を示す文集合のリストを $A_i \equiv \{A_{i,1}, A_{i,2}, ..., A_{i,l}\}$ とする．このとき，m_i に対して， l 個の対応文集合が存在することになる．コーパスでは，これらに基づいた， 人手による short,long の要約文が用意されている．

5.2.2 評価指標

NTCIR4 ― TSC3 では評価指標として，人手による要約評価指標と，自動要約評価指標と 2 種類あり，人手によらない自動要約評価指標には正解率と被覆率とがある．本稿では要約の精度評価としてこの自動評価を採用する．本稿では，TSC3 でも比較を行った正解率，被覆率に加え，要約の冗長性比較の値として定義される重要文冗長性 [53] の評価を行う．

重複情報削減の有効性は重要文冗長性に対して発揮されるのであるが，要約として空文字列を与えることで重要文冗長性を 0 にすることができる．重複情報を削減しつつ，必要な情報もカバーしていることを示すために正解率，被覆率の値も評価する．

正解率

正解率は，システム出力の文のうち，正解要約に含まれる文の割合である．次の式で定義される．

$$\text{正解率} = \frac{m}{h} \tag{5.10}$$

h は制約充足問題 [53] を解いて得られた，正解要約を生成するために必要な最小の文数であり，m はシステムが出力したうちの正解文数である．

被覆率

被覆率 [53] は，システム要約がどの程度正解要約に近いか，その冗長性を考慮しつつ測る値である．参照要約の i 番目の文 m_i に対応する元テキストのリストを A_i と表し，システム要約の文集合を E として表す．このとき文 m_i に対する評価値 を次に定義する

$$e(i) = \max_j \left(\frac{|A_{i,j} \cap E|}{|A_{i,j}|} \right) \tag{5.11}$$

関数 $e(i)$ は参照要約の i 番目の文に対する対応文集合 $A_{i,j}$ のいずれかを完全な形で出力した場合には 1，部分的に出力した場合には $|A_{i,j}|$ に応じて部分点を与える関数である．この関数と参照要約の文数 n を用いて被覆率を以下の式で定義する．

$$\text{被覆率} = \frac{\sum_{i=1}^{n} e(i)}{n} \tag{5.12}$$

重要文冗長性

重要文冗長性は，システム要約が正解要約を充足する情報のうち冗長となる文の量を定量的に表した数値である．

参照要約の m_i に対応する文集合 $A_{i,j}$ の和集合を L_i とする

$$L_i \equiv \bigcup_j A_{i,j} \tag{5.13}$$

冗長性を考慮せずにシステム要約が正解要約を充足する文集合は $E \cap L_i$ と表せる．L_i を充足する E の部分集合の中で要素数が最小のものを $|S_i^{min}|$ とする．これは，E の全ての文を用いなくても $|S_i^{min}|$ の文を用いるだけで

$e(i)$ と等価な情報を得ることが出来ることを示している．ここから，m_i に関して正解要約でかつ $e(i)$ を得るために貢献していない，つまり冗長な文の数は，次の式となる．

$f_i(E) = |E \cap L_i| - |S_i^{min}|$ ここから，重要文冗長性を以下の式で定義する．参照要約の数が n であるとする．

$$\text{重要文冗長性}\,(E) = \frac{\sum_{i=1}^{n} f_i(E)}{n} \tag{5.14}$$

重要文冗長性は 0 以上であり，他の指標とは違い，上限値はない．

5.2.3 ROUGE

要約のもう一つの評価手法として，DUC [1] で評価用に用いられる ROUGE [17] という指標がある．ROUGE の元となった BLEU [27] は，もともと機械翻訳の翻訳結果の評価を行うための手法であり，要約の評価指標としても用いられるようになった．

要約評価の場合，再現率が重要となるため，BLEU のように精度による評価は適当ではなく，また，要約はできるだけ短く生成したいが，要約文が短い場合の補正が必要などの問題点があったため，これらを改善した式として ROUGE が提案された．

$$C_n = \frac{\sum_{C \in \{ModelUnits\}} \sum_{n-gram \in C} Count_{match}(n-gram)}{\sum_{C \in \{ModelUnits\}} \sum_{n-gram \in C} Count(n-gram)} \tag{5.15}$$

人間が作成した（正解）文書をシステムの生成した要約がどの程度カバーしているかを計算する式である．この式は基本形の ROUGE-N のものであり，システム要約の N-gram が正解要約の N-gram をカバーする割合である．

ROUGE には他のバージョンも存在し，係り受け関係を考慮し，共起関係を用いた ROUGE-S や，それに加えて uni-gram を追加した ROUGE-S などもある．

[1] Document Understanding Conferences http://duc.nist.gov/

5.2.4 実験方法

TSC3 に含まれる 30 のニュースクラスタについて，short 及び long の指定文字数以内の要約文をそれぞれ生成する．こうして出来たシステム要約について， 正解率，被覆率，重要文冗長性による評価値を算出する．

今回の実験でわれわれは提案手法と他の複数文書要約手法との比較を行った．比較手法は TSC3 のベースラインとして挙げられている LEAD, TF-IDF と， アラインメント方式の代表として ClustalW を用いた．これらに加え TSC3 に参加した 8 つのシステム (SOUKEN, CRLNYU, smlab, MOGS, forest, DBLAB, UEC, UYDI) との比較を行った．

正解率，被覆率については working note [11] に掲載されている値を用いた．一方，重要文冗長性については，今回改めてシステム要約から計算した．DBLAB， MOGS， forest の 3 つのシステムについては，NTCIR4 のタスクとして提出したシステム要約 [36, 25, 23] を提供していただき，そこから計算したものである．表 5.3 の値は working note [11] に掲載されたものに，提案手法 (TA) および ClustalW の結果を加えたものである．図表の値は 30 のニュースクラスタにおける評価値の平均値である．

追加として，類似一致長の下限 e を 16,18,20,22,24,26,28,30,32 と変化させた場合の 正解率，被覆率，重要文冗長性の変化も計算した．それぞれ，表 5.5, 表 5.6, これらをグラフ化したものが図 5.10,5.11,5.12 である．

5.3 結果の比較と考察

表 5.3 ， 図 5.8 ， 図 5.9 に示すとおり TSC3 による評価指標の正解率，被覆率では提案手法は他の手法に劣る．他手法では 文の「重要さ」を考慮した文選択を行うが，提案手法ではこのような文選択をしないため，このような結果になっていると考えられる．

提案手法は自然言語を単純な記号列としてしか扱わず，品詞の区別や係り受け解析など文法的知識を必要としていないように，深い言語的知識を用いていないが，上記実験結果が示すように，ある程度の質の要約文を生成することができる．深い言語知識を用いず，単純な文選択だけでよい理由として，対象となるコーパスが電子的テキストであり，なおかつ同一の話題に関する新聞記事の特性が関係するものと思われる．このようなコーパスの傾向として，はじめに述べたとおり，記事見出しのような類似した表現の並び，類似した言い回しが頻出し，そのような一文を読んだだけでトピックがわかる場合が多いからと考えられる．

	short		long	
	coverage	precision	coverage	precision
SOUKEN(a)	0.315	0.494	0.355	0.554
SOUKEN(b)	0.372	0.591	0.363	0.587
CRLNYU(a)	0.222	0.314	0.313	0.432
CRLNYU(b)	0.293	0.378	0.295	0.416
smlab	0.328	0.496	0.327	0.535
MOGS	0.283	0.406	0.341	0.528
forest	0.329	0.567	0.391	0.68
DBLAB	0.308	0.505	0.339	0.585
UEC	0.181	0.275	0.218	0.421
UYDI	0.251	0.476	0.247	0.547
LEAD	0.212	0.426	0.246	0.539
TF-IDF	0.292	0.497	0.325	0.604
ClustalW	0.208	0.340	0.247	0.536
提案手法	0.267	0.404	0.330	0.545

表 5.3: NTCIR4 TSC3 における指標

	short		long	
	ROUGE	重要文冗長性	ROUGE	重要文冗長性
DBLAB	0.481	0.349	0.495	0.331
MOGS	0.393	0.270	0.455	0.336
forest	0.459	0.351	0.507	0.499
LEAD	0.368	0.236	0.395	0.395
TF-IDF	0.414	0.164	0.448	0.214
ClustalW	0.350	0.087	0.416	0.175
提案手法	0.438	0.077	0.453	0.129

表 5.4: 追加指標

e	precision	coverage	重要文冗長性
16	0.451	0.253	0.101
18	0.410	0.256	0.084
20	0.422	0.265	0.064
22	0.464	0.266	0.055
24	0.404	0.267	0.077
26	0.430	0.214	0.079
28	0.430	0.214	0.079
30	0.438	0.227	0.101
32	0.430	0.214	0.079

表 5.5: e に対する性能 (short)

e	precision	coverage	重要文冗長性
16	0.564	0.277	0.125
18	0.554	0.274	0.105
20	0.546	0.274	0.102
22	0.563	0.275	0.133
24	0.545	0.330	0.129
26	0.549	0.237	0.099
28	0.549	0.237	0.099
30	0.575	0.250	0.092
32	0.549	0.237	0.099

表 5.6: e に対する性能 (long)

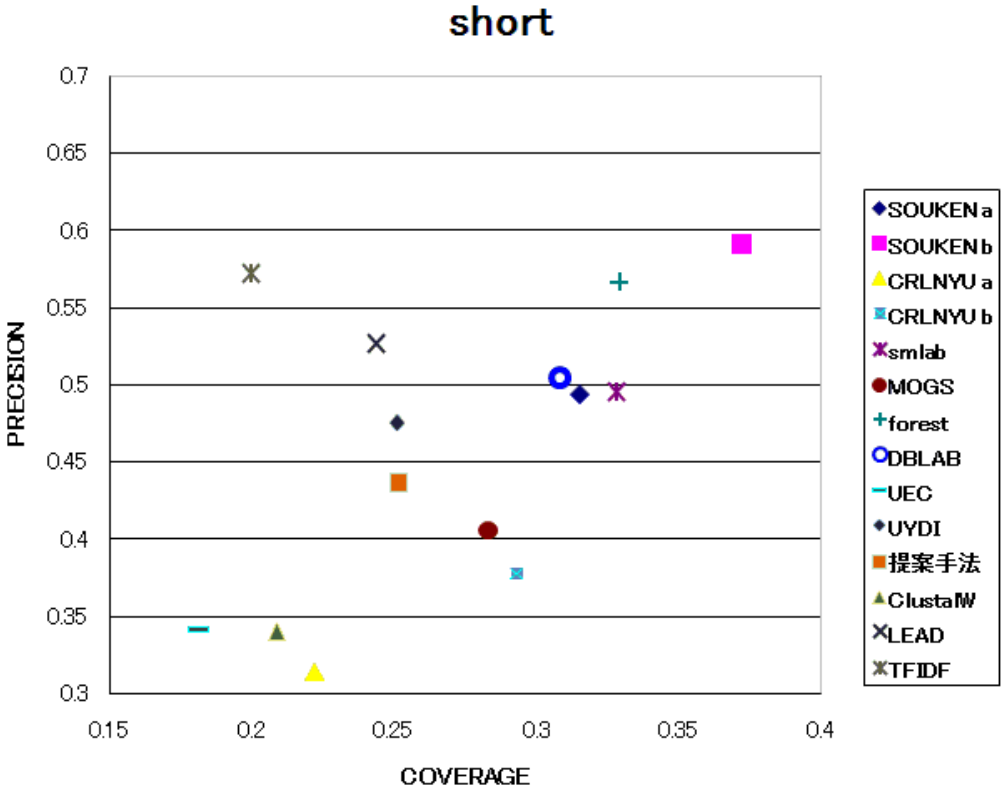

図 5.8: NTCIR4 TSC3 における指標 (short)

表 5.5，表 5.6，図 5.10-5.12 に示す e と数値との関係を示している．重要文冗長性は e の値に影響を受ける． この理由であるが，e が短いとより多くの文を持つ文クラスタが多数生成される可能性が高くなる．類似した文が集積されることで重要文冗長性を下げることが期待できるが，同時に，クラスタ数が増えることで本来は選択されるべきではない文が入ってくる．

e が長いとクラスタの数は平均的には小さく，要素文の数も少なくなる．類似した文の集積が減ることで重要文冗長性は上がると予想されるが，クラスタ数が減ったために冗長性を増す”余計”な文が選択されることがなくなる．e がクラスタ数とクラスタの大きさを同時に制御しているために，単純な相関が生じない．

正解率，被覆率が e の変化に対して比較的安定している理由は，数値を向上させる文の幅が広いことである．例えば正解率の場合は，要約要

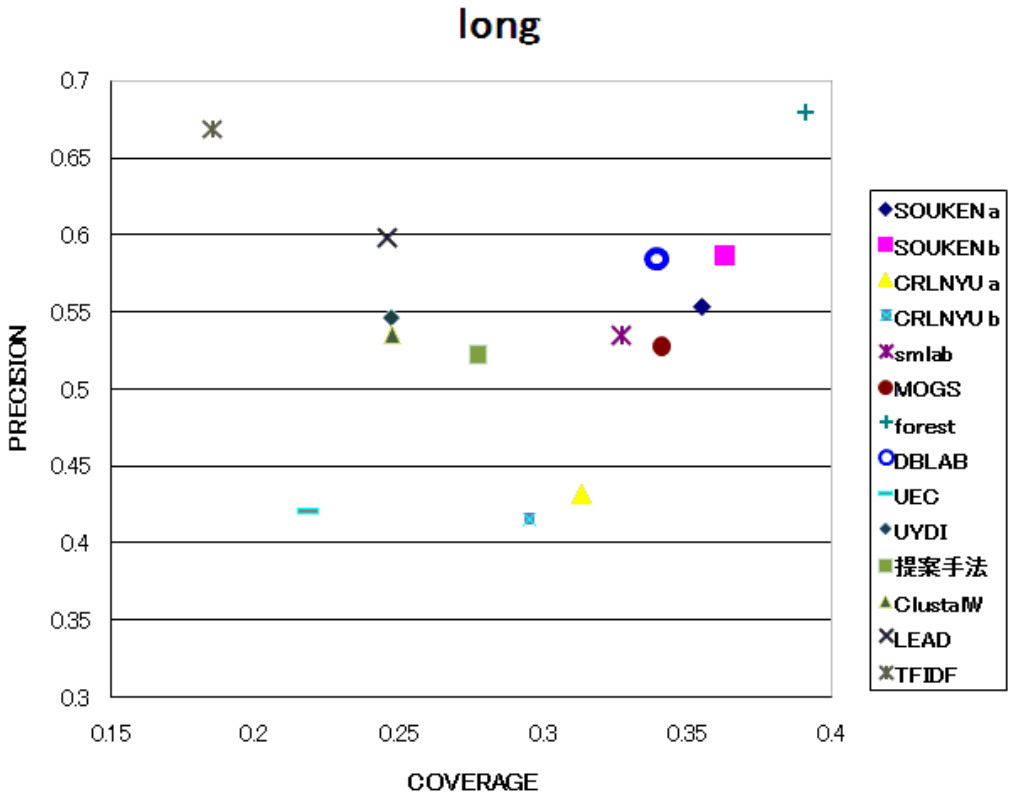

図 5.9: NTCIR4 TSC3 における指標 (long)

素の全ての文を選択することで最高の数値を出すことはできる．重要文冗長性の増減は必ずしも正解率，被覆率の向上，低下に直接結びつかないのである．

重要文冗長性について， 表 5.4 に示すとおり，提案手法が最も低い冗長性を実現した．提案手法とそれ以外の手法との差は， t 検定で 1 % 以下の優位水準の棄却域に十分に入る大きな差であり，提案手法が最も良い精度を出している．これにより提案手法の目的である冗長性の検出と，有効性が示せた．また，ClustalW でも TA と同様に 他の手法に比べ十分に重要文冗長性を低く抑えられている．これは，他の要約手法に対し，提案手法のように文字列の並び順を考慮した類似度の方が，自然言語の冗長性除去においてはより有効であることを示していると考えられる．理由として，代替可能な文集合は高い確率で非常に類似した部分文を持っ

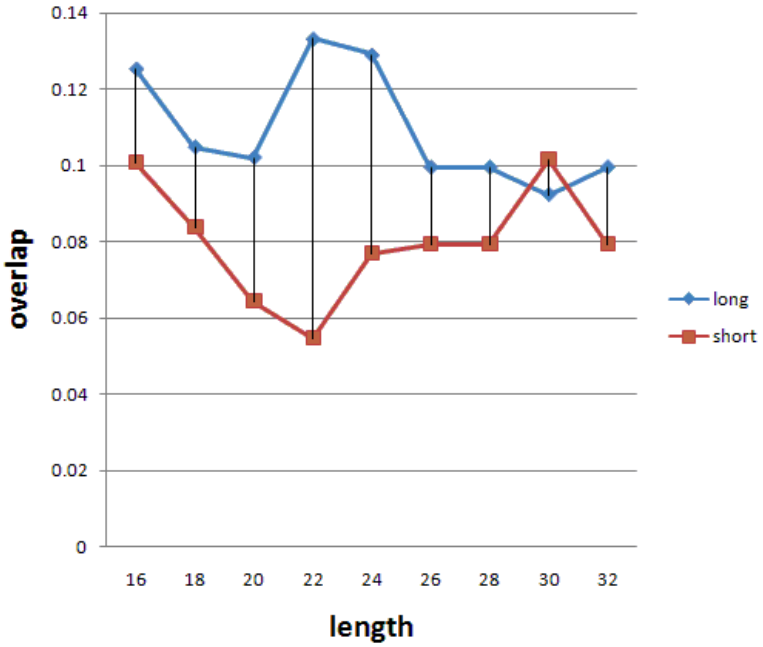

図 5.10: e に対する重要文冗長性

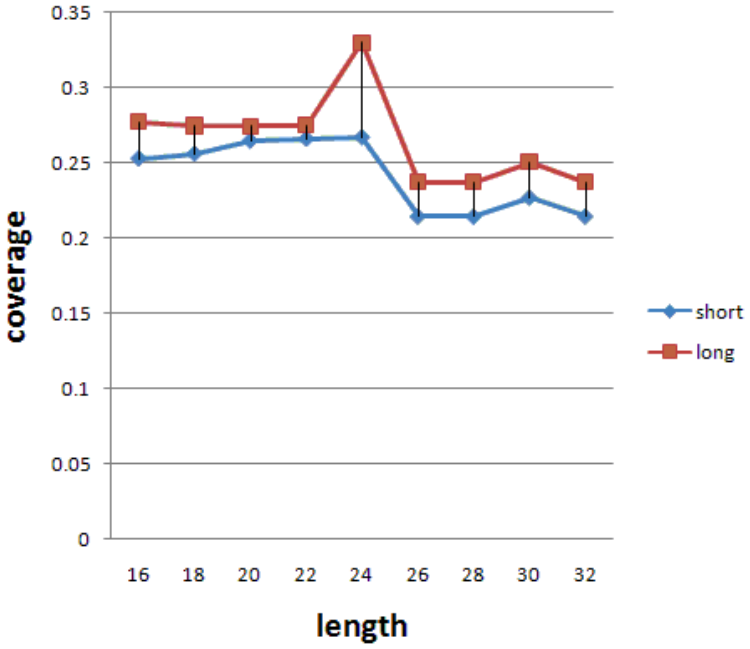

図 5.11: e に対する被覆率

ているためである．少なくとも，今回の評価用コーパスではこのような傾向があった．

もちろん，類似部分以外で重複しない情報を持っている可能性はある．既存手法の場合，類似部分以外の情報量が強く効いた文が要約要素として追加選択され，結果として重要文冗長性を増やしていると思われる．

他の手法では，文書を代表する文を選択し，文の重要性を考慮して要約を作成しようとするが，提案手法の場合はこのような部分文を検出し，同様の領域を持つ集合から 1 文だけを選択することで，冗長な情報を出来る限り排除し，結果的に重要文冗長性を低く抑えていると思われる．これは正解率を犠牲にする選択法であり，要約の目的に応じて使い分ける必要がある．

冗長性を極めて低く抑えることのできる本手法は，十分に短い要約文を作成したい場合に役立つ．PC など充分な量の情報表示が出来るデバイスではこの効果は顕著ではないが，特に携帯電話に代表される，表示できる文字数に限りがある小型デバイスなどでは，十分に短い文章で情報を記述しなければならない．

現在，Web 上には情報が氾濫していると言われているが，実際には情

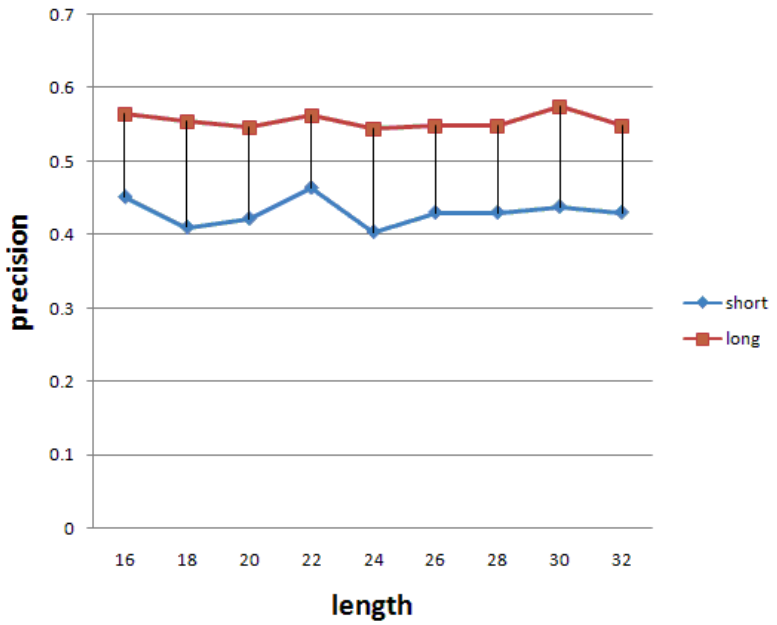

図 5.12: e に対する正解率

報が氾濫している様に見えて異なる表現を使ってほとんど同一の意味を表している文章も多く存在する．しかも，単一の文書そのものが冗長であるのではなく，部分的に冗長性が存在するのであり，それ以外の部分は新規情報である場合も多い．提案手法は，このような部分的重複情報の特定 [50] にも応用できる．

5.4 計算量比較

アラインメント計算量比較のために ClustalW との比較を示す．図 5.13 にその時のデータ量と実行時間との関係を示す．

ClustalW の場合はデータ量の増加に伴って計算量が増大するが提案手法 TA ではデータ量が増えても計算量は指数的には増加しない．文書の長さ（単語数）を N， 文書の数を D とした場合，理論的には提案手法 TA の計算量のオーダーは $NDlog(ND)$ となる．

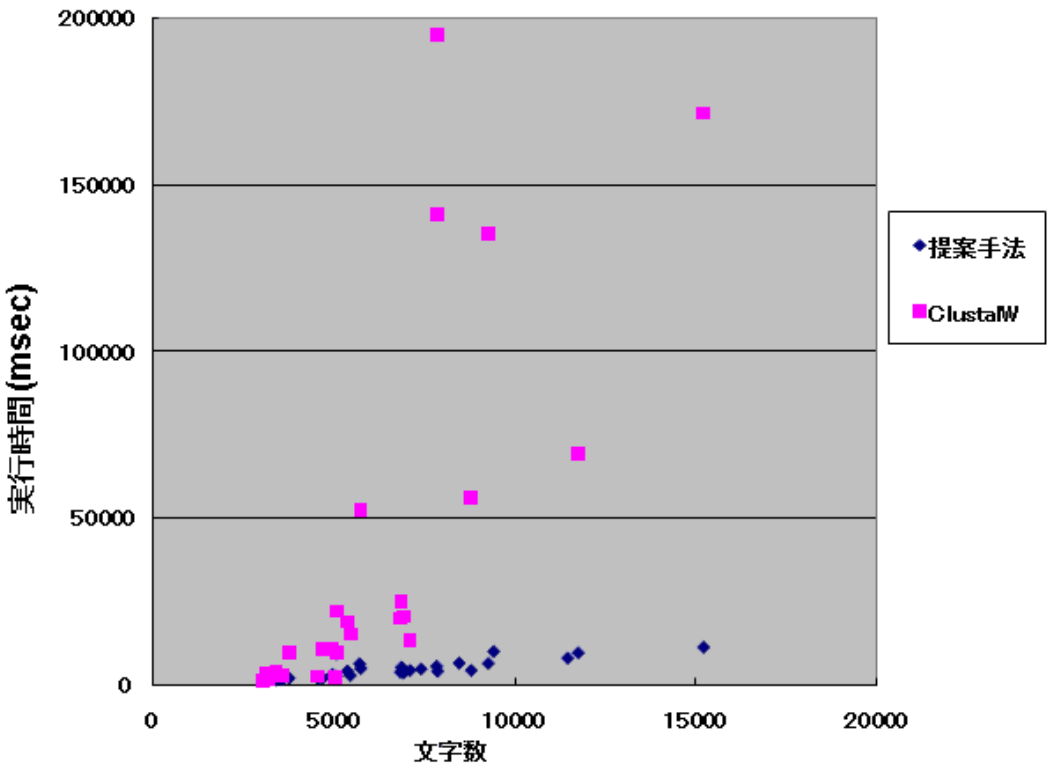
200000
150000
100000
50000
0
実行時間(msec)
0
5000
10000
15000
20000
文字数
提案手法
ClustalW

図 5.13: 実行時間

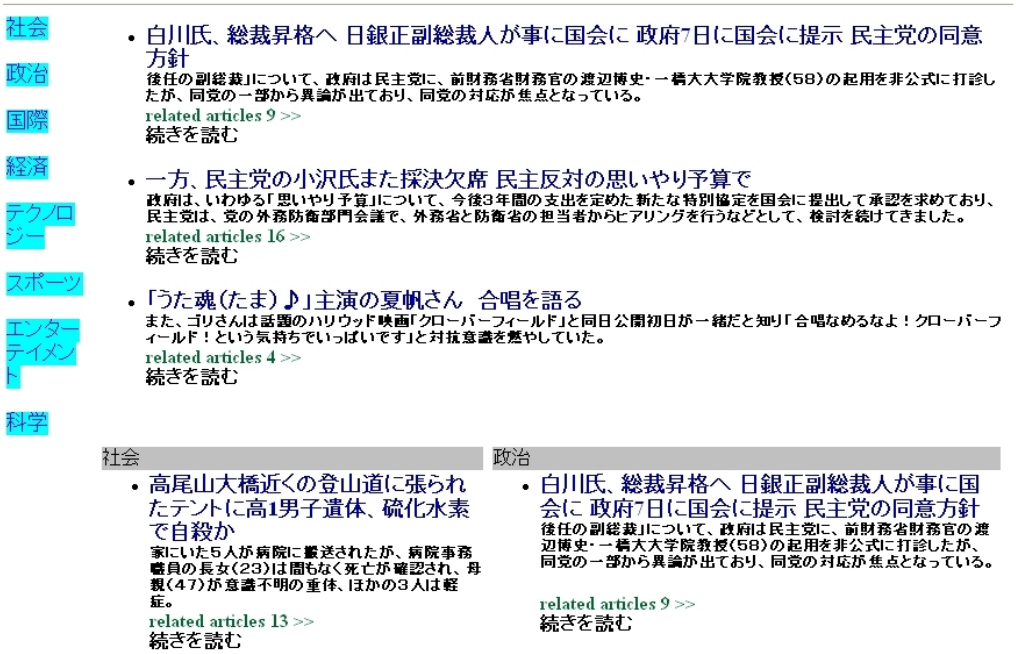

図 5.14: トップページ

5.5 UpdateNews：ニュース記事の要約システム

5.5.1 システム概要

筆者らは，ニュース記事の収集と分類および要約を行う UpdateNews [33] システムを構築した．このシステムは，Web 上に公開されているニュース記事を収集し，カテゴリごとに分類して提供する．図 5.14 に，UpdateNews のトップページを示す．

収集した記事について，同一トピックに言及している記事ごとにクラスタ作成し，各クラスタごとに要約を生成する.（図 5.15）

ニュース記事は 8 カテゴリ（社会，政治，国際，経済，テクノロジー，スポーツ，エンターテイメント，科学) に分類される．この 8 カテゴリは，クローリング対象の各ニュースサイトで事前に分類され与えられているメタデータによって分類している．収集した記事について，同一の物事について言及している記事ごとに分類されるようにクラスタリングを行い，こうしてできたトピックごとに，その複数記事を元文書とした要約生成を行う．本システムは ニュースクローラ，クラスター分類器（生成

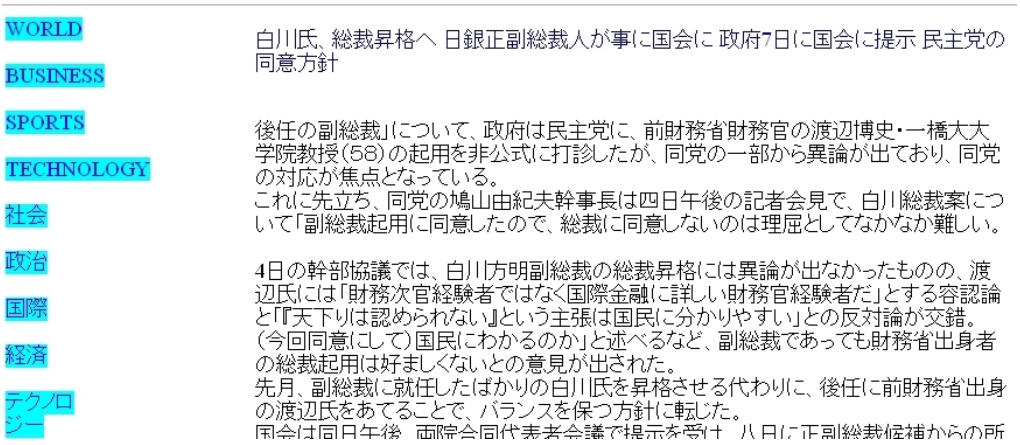

図 5.15: 要約文書表示ページ

器），要約器 （図 5.16） によって構成される．

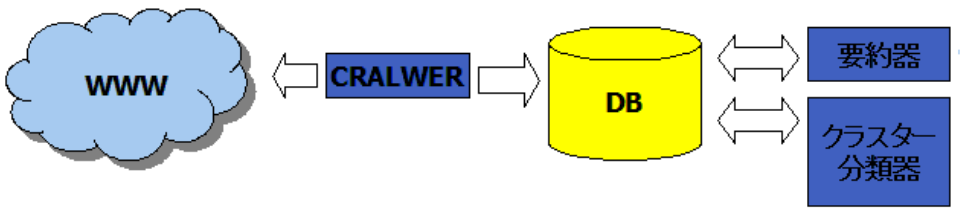

図 5.16: システム設計

5.5.2 収集対象

定期的にクローラが各ニュースサイトの更新情報を示す RSS フィードをチェックし，更新があったコンテンツを取得する．(図 5.17)

RSS フィード 内で単一の記事は <item> タグによって示される．それぞれ各タグが示す情報は以下の通りである．

1. <title> 記事タイトル
2. <link> 記事 URL

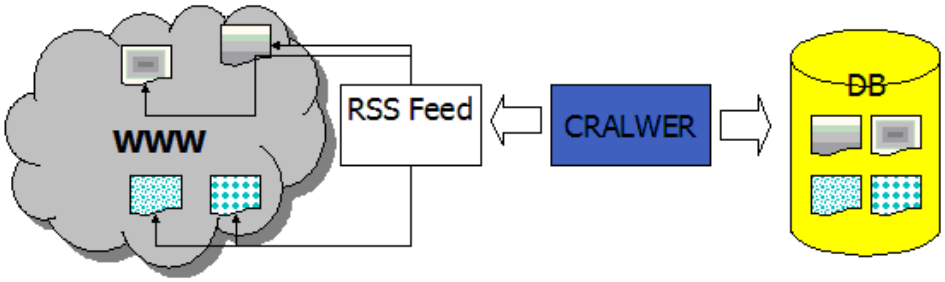

図 5.17: RSS クローラ

3. <date> 記事の時刻

4. <description> 記事見出し

記事の URL が示す html ファイルから html タグや広告など記事に直接関係ない情報を削除し，本文のみを抜き出す．この方法は，各ニュースサイトごとに，<div class=”main boby”> など，本文を示すタグが存在するため，このタグの内側のテキストタグを取得することにより，記事本文の抽出が可能である．

5.5.3 クラスタリング

システムは各記事を適切なトピックに割り当てる．文書分類には様々な関連研究 があるが，本システムでは Vector space model を用い， 各単語の値を inverse document frequency によって重み付ける．各文書に含まれる語の値を $\{w_1, w_2, ..., w_n\}$ とする． w_i の document frequency を $df(w_i)$ とする．これはコーパス全体 A_{all} 中の出現頻度である．$tf(w_i, a)$ は，単語 w_i の記事 a 中での出現頻度を表す．このようにして，記事 a を以下のように n 次元のベクターで表現する． $\overrightarrow{a} = (f_1, f_2, ..., f_n)$

このとき，

$$f_i = tf(w_i, a) \times idf(w_i) \tag{5.16}$$

$$idf(w_i) = log\left(\frac{|A_{all}|}{df(w_i)}\right) \tag{5.17}$$

あるトピックに属する記事の集合を $T \equiv \{b_1, b_2, ..., b_{|T|}\}$ とすると．このトピック T と記事 a の類似度を以下のように cosine 距離によって計算する．

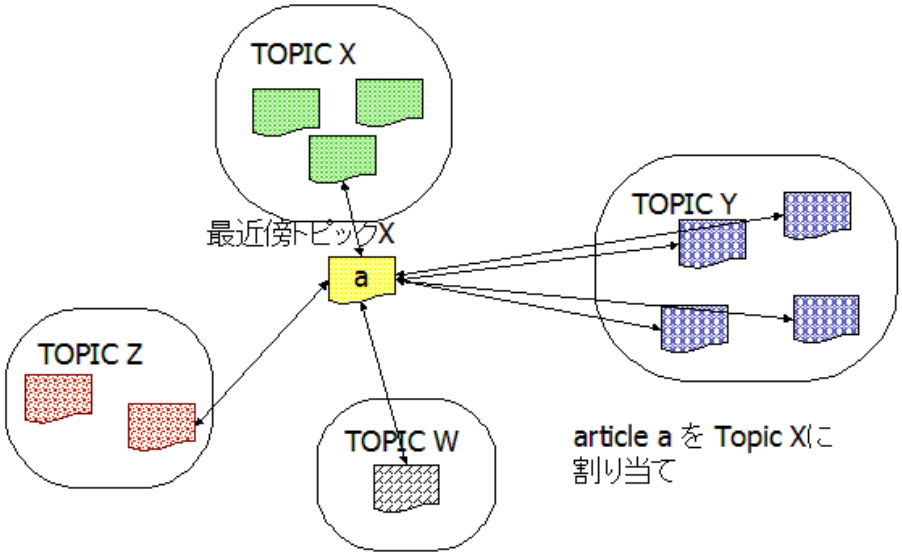

図 5.18: クラスタ生成 例

$$sim(T,m) = max_{b \in T} \frac{\overrightarrow{a} \cdot \overrightarrow{b}}{\|\overrightarrow{a}\| \times \|\overrightarrow{b}\|}$$

$\|\overrightarrow{a}\|$ はベクトルの大きさである．文書 a を，閾値以上で最も大きい類似度を示したトピックに割り当てる．閾値以上のトピックがなければ a のみの新しいトピックを作る． (図 5.18)

5.6 システム運用状況

図 5.19 に時間別，日別の収集記事件数を示す．これらすべての記事についてクラスタリングを行っている．

また，表 5.7 に日別の要約生成件数を示す．

5.7 マシンスペック

本システムは，現在表 5.8 に示すマシンで運用している．このスペックのサーバ 2 台分の計算機資源でシステムは実現可能である．図 5.16 中の

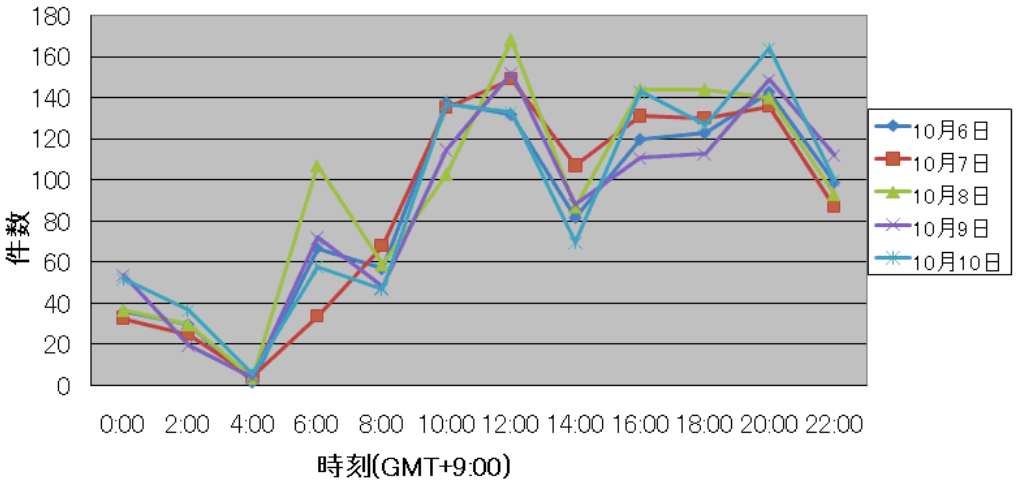

図 5.19: 収集記事数

	件数
2008-10-06	311
2008-10-07	312
2008-10-08	362
2008-10-09	347
2008-10-10	311

表 5.7: 要約生成数

vendor_ id	AuthenticAMD
cpu family	15
model	65
model name	Dual-Core AMD Opteron(tm) Processor 2218 HE
stepping	3
cpu MHz	2600.000
cache size	1024 KB
flags	fpu tsc msr pae mce cx8 apic mca cmov pat pse36 clflush mmx fxsr sse sse2 ht syscall nx mmxext fxsr_ opt lm 3dnowext 3dnow pni cx16 lahf_ lm cmp_ legacy svm cr8legacy ts fid vid ttp tm stc
bogomips	5202.00

表 5.8: マシンスペック

Cralwer と クラスタ分類とを一台が担当し，データベースサーバと要約生成をもう一台が担当するという構成である．

第6章　結論

6.1　本博士論文のまとめ

本稿では局所的類似性を検出する手法の提案と，評価を含めた応用問題への適応に取り組んだ．

まず Local-Multiple-Text Alignment の考えに基づく重複情報の削減法を提案した．提案手法は冗長性除去の観点において有効な手法であり，任意の類似領域の特定による冗長性除去と，複数文書要約への新しい可能性を示せた．

もう一つの問題としてコピー文字列検知に基づいた splog フィルタリング手法を提案した．提案手法は，IDF に基づいた重み付きコピー文字列長を導入し，そのコピー率に基づいて，splog 判定を行う．重み付きコピー文字列長を効率良く計算するために，suffix array と動的計画法を用いたアルゴリズムを提案し，また，splog フィルタリングの性能を評価するためのコーパスを作成して，提案手法の性能を評価した．

本稿の提案手法は，事前の予備知識なしで，類似情報の検出することができる．特に品詞や文法など言語別に生じる特徴などを意識することなく，統計的特徴のみで十分である

この利点は逆にデメリットにもなる．まず，十分な規模のコーパスを準備する必要があるということと，もう一つ，計算量削減のための様々な工夫を加えたとはいえ，まだまだ計算コストが非常に大きいという問題がある．

6.2　今後の課題と展望

複数文書要約の提案手法は，任意の複数文書に出現する任意の類似領域を検出するためにも使えるため，多くの応用が考えられる．提案手法は自然言語を記号列としてしか扱わず，品詞の種別や係り受け解析など

文法的知識を必要としないため，比較的容易に多言語文書へ応用も期待できる．

提案手法は任意の類似表現を検出するため，冗長性の削減だけではなく，類似情報の集積にも用いることができる．類似した部分表現のカウントによって単語レベルの統計だけでは見えてはこない傾向を調べられる．

しかしながら本稿の手法はシンプルなアルゴリズムであり，パラメータの最適化なども含め，場合に応じた様々なバリエーションが考えられる．今後は，ニュース記事など充分に文体が整った文書ではなく，blog 記事など表現の多様性が大きい文書群，チャットや電子掲示板，SNS(Social Networking Service) などユーザ間のコミュニケーションが介在するコンテンツ， アンケート集計や商品レビューの自動要約，また複数言語への応用も視野に入れている．

類似領域の検出ではなく， 逆にオリジナルコンテンツの検出を行うという使い方も考えられる．しかしながら,（コピーが全く存在しない）ユニークな情報というものはほとんど存在せず，また，本質的にはコンテンツの 公表時間も考慮して，時間的に後の方がコピーされたものであると考えることが正しいが，提案手法ではコンテンツの時間を考慮していないため，このような詳細な問題設定までは踏み込んではいない．

今回は，オリジナルコンテンツ検出の有効なアプリケーションと有効な評価方法が設計できなかったため取り組んでいないが，テキストストリームからの新情報検出 (novelty detection) [1] [1] や，商品などのレビュー要約などへの応用が有望である．評価は行っていないが，レビュー記事の差分から要約を作るアプリケーションシステムを作成した．

「局所的類似性」をみつける技術を用いることによって得られる理想的情報システムの一例は，あるユーザのコンテンツ閲覧記録から，当該ユーザ（このブラウザ）にとっての既知情報を，明示的にハイライト，または隠蔽するブラウザ，などである．これによってユーザはもうすでに知っている情報がどの部分にあり，まだ知らない情報がどの部分にあるのかを知ることができ，既知情報を何度も閲覧することがなくなり情報取得効率の向上が期待できる．

splog filter の提案手法によって高いフィルタリング性能を実現できたが，未だに検出できない splog が多く残っている．今後は，フィルタリング性能を向上するための，手法の改良を行うことを計画している．普遍的な文字列の生成確率に基づいた**全ての**コンテンツを用意できるのであ

[1] TDT http://www.nist.gov/speech/tests/tdt/

れば，それとの情報量比較によってフィルタリングができる可能性はあるが，そのような超巨大なコーパスを用意することが現実的に困難なだけではなく，存在を仮定することも困難である．文字列の生成確率が時間と共にどのように変化していくのか不明だからである．現実的には，今回のように小規模なコーパスを用いて，コピーの量によって splog の判定を行う．

また詳しく調べると，splog はユーザに見える部分だけではなく，html ソースのレベルにおいても特徴的な記述が多いため，これらの部分も用いることによってフィルタリング性能を向上できる可能性がある．

次に，splog では，特有の言い回しが頻出することが多いため，これを特定できればより簡単に splog を検知することが可能になる．しかし，blog においても，ブログパーツと呼ばれる，手動でコピーして記述する文字列が多数存在する．本研究で blog を splog と誤った多くのケースで，ブログパーツを splog のコピー文字列として検出していた．今後はこれらの自動的な分離を目指す．

現在までには（人間の手作業を含めた）半自動での splog filter を開発することができたが，今後，splog の傾向が変化する可能性はある．より，確実で効率的な手法の研究の余地がある．

謝辞

本論文を執筆するにあたり，多くの方々のご指導とご協力を賜りました．ここにお世話になった方々への感謝の意を表します．

まず何より，総合研究大学院大学複合科学研究科情報学専攻における研究生活を支えていただきました指導教員である国立情報学研究所の高須淳宏教授に心より感謝いたします．わたくしの研究テーマについて，様々な助言と，研究に付随する総合的なご指導を重ねてくださいました．本論文を完成させられたのも，ひとえに高須教授のご指導によるものです．心より厚く御礼申し上げます．

アドバイザを務めてくださいました影浦峡准教授と，安達淳教授に深く感謝いたします．影浦准教授と，安達教授には，それぞれご専門の立場から，本研究に関する有益な様々なご意見を頂戴いたしました．厚く御礼申し上げます．

NTCIR4-TSC3 に関するご質問にご回答いただき，要約結果を提供していただいた，東京大学の岡崎直観氏，北海道大学の吉岡真治准教授，横浜国立大学の森辰則教授に，御礼を申し上げます．特に評価指標に関する貴重なご助言を度々いただいた吉岡真治准教授には大変お世話になりました．

Splog Filter に関するご質問にご回答いただきました島根県立大学の石田和成准教授，九州大学の竹田正幸教授，同 成澤和志氏には，深く御礼申し上げます．

Splog Filter に関して，ベンチマークデータが作成できたのは，有限会社アジマッチの中本賢吾社長の多大なご協力によるものです．ありがとうございました．

株式会社ゴーガの小山文彦社長には，本研究技術のさまざまな形での製品への導入を企画してくださり，心から感謝いたします．

さらに，研究成果の社会への還元について，非常に有益なご助言をいただいた大向一輝 博士にも深く感謝いたします．splog filter の製品への導入と実用化に関しては大向博士のご助言があったからこそです．

内山幸樹社長，成瀬功一郎取締役，セーヨー・サンティ氏，浅野弘輔氏，嶋村壽晃氏をはじめとしました株式会社ホットリンクの皆さま，並びに東京大学の福原知宏氏，同 松尾豊准教授へ，本技術の採用と，製品化への尽力に，ご協力いただき深く感謝いたします.

参考文献

[1] Allan, J.: *Topic Detection and Tracking: Event-based Information Organization.*, Kluwer (2002).

[2] BAEZA-YATES, R.: Efficient Text Searching, *Ph.D. thesis, Dept. of Computer Science, University of Waterloo* (1989).

[3] Barzilay, R., McKeown, K. and Elhadad, M.: Information fusion in the context of multi-document summarization, *In Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pp. 550–557 (1999).

[4] Brin, S. and Page, L.: The anatomy of a large-scale hypertextual web search engine, *In Proceedings of the seventh international conference on World Wide Web 7*, pp. 107–117 (1998).

[5] Carbonell, J. and Goldstein, J.: The use of MMR,diversity-based reranking for reordering documents and producing summaries, *In Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 335–336 (1998).

[6] Corpet, F.: Multiple sequence alignment with hierarchical clustering, *Nucleic Acids Research*, Vol. 16, No. 22, pp. 10881–10890 (1988).

[7] Frith, M. C., Hansen, U., John, S. L. and Weng, Z.: Finding functional sequence elements by multiple local alignment, *Nucleic Acids Research*, Vol. 32, No. 1, pp. 189–200 (2004).

[8] Fujimura, K., Inoue, T. and Sugisaki, M.: The EigenRumor Algorithm for Ranking Weblogs, *2nd Annual Workshop on the Weblogging Ecosystem: Aggregation, Analysis and Dynamics, WWW 2005* (2005).

[9] Goldstein, J., Mittal, V., Carbonell, J. and Kantrowitz, M.: Multi-document summarization by sentence extraction, *In Proceedings of the ANLP/NAACL Workshop on Automatic Summarization*, pp. 40–48 (2000).

[10] Hatzivassiloglou, V., Klavans, J. L., Holcombe, M. L., Barzilay, R., Kan, M. and McKeown, K. R.: SIMFINDER: A flexible clustering tool for summarization, *In NAACL Workshop on Automatic Summarization, Association for Computational Linguistics*, pp. 41–49 (2001).

[11] Hirao, T., Okumura, M., Fukushima, T. and Nanba., H.: Text Summarization Challenge 3 -Text Summarization Evaluation, *NTCIR Workshop4-Working Notes of the Fourth NTCIR Workshop Meeting* (2004).

[12] Kolari, P., Finin, T. and Joshi, A.: SVMs for the blogosphere: Blog identification and splog detection, *AAAI Spring Symposium on Computational Approaches to Analyzing Weblogs* (2006).

[13] Kolari, P., Java, A. and Finin, T.: Characterizing the Splogosphere, *In Proceedings of the 3rd Annual Workshop on Weblogging Ecosystem: Aggregation, Analysis and Dynamics, 15th World Wid Web Conference* (2006).

[14] Kolari, P., Java, A., Finin, T., Oates, T. and Joshi, A.: Detecting Spam Blogs: A Machine Learning Approach, *In Proceedings of the 21st National Conference on Artificial Intelligence* (2006).

[15] Kurtz, S.: Approximate String Searching under Weighted Edit Distance., *In Proceedings of Third South American Workshop on String Processing*, pp. 156–170 (1996).

[16] Li, K.-B.: ClustalW-MPI: ClustalW analysis using distributed and parallel computing, *Bioinformatics*, Vol. 19, No. 12, pp. 1585–1586 (2003).

[17] Lin, C.-Y.: ROUGE: A Package for Automatic Evaluation of summaries, *In Proceedings of the ACL-04Workshop: Text Summarization Branches Out*, pp. 74–81 (2004).

[18] Lin, Y. R., Chen, W.-Y., Shi, X., Sia, R., Song, X., Chi, Y., Hino, K., Sundaram, H., Tatemura, J. and Tseng, B.: The splog detection task and a solution based on temporal and link properties, *In Poceedings of the 15th Text REtrieval Conference* (2006).

[19] Lin, Y. R., Sundaram, H., Chi, Y., Tatemura, J. and Tseng, B. L.: Splog detection using self-similarity analysis on blog temporal dynamics, *In Proceedings of the 3rd International Workshop on Adversarial Information Retrieval on the Web*, pp. 1–8 (2007).

[20] Manber, U. and Myers, G.: Suffix arrays: a new method for online string searches, *In Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, pp. 319–327 (1990).

[21] McKeown, K., Barzilay, R., Evans, D., Hatzivassiloglou, V., Schiffman, B. and Teufel, S.: Columbia multi-document summarization: Approach and evaluation, *In Proceedings of the Document Understanding Conference* (2001).

[22] McKeown, K. R., Barzilay, R., Evans, D., V. Hatzivassiloglou, J. L. K., Sable, C., Schiffman, B. and Sigelman, S.: Tracking and summarizing news on a daily basis with Columbia's Newsblaster., *In Proceedings of the Human Language Technology Conference*, pp. 280–285 (2002).

[23] Mori, T., Nozawa, M. and Asada, Y.: Multi-Answer-Focused Multi-Document Summarization Using a Question-Answering Engine, *Transactions on Asian Language Information Processing*, Vol. 4, No. 3, pp. 305–320 (2005).

[24] Narisawa, K., Inenaga, S., Bannai, H. and Takeda, M.: Efficient Computation of Substring Equivalence Classes with Suffix Arrays, *In Proceedings. 18th Annual Symposium on Combinatorial Pattern Matching*, pp. 340–351 (2007).

[25] Okazaki, N., Matsuo, Y. and Ishizuka, M.: TISS: An Integrated Summarization System for TSC-3, *Proceeding of NTCIR-4* (2004).

[26] Page, L., Brin, S., Motwani, R. and Winograd, T.: The PageRank citation ranking: Bringing order to the Web, *http://google.stanford.edu/∼backrub/pageranksub.ps* (1998).

[27] Papineni, K., Roukos, S., Ward, T. and Zhu., W.: BLEU: a Method for Automatic Evaluation of Machine Translation, *IBM Research Report* (2001).

[28] Radev, D. R., Blair-Goldensohn, S., Zhang, Z. and Raghavan, R. S.: NewsInEssence: A system for domain-independent, real-time news clustering and multi-document summarization, *In Human Language Technology Conference*, pp. 1–4 (2001).

[29] Radev, D. R., Jing, H. and Budzikowska, M.: Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies, *In Proceedings of the ANLP/NAACL Workshop on Automatic Summarization* (2000).

[30] Salvetti, F. and Nicolov, N.: Weblog classification for fast splog filtering: A url language model segmentation approach, *In Proceedings of the Human Language Technology Conference of the NAACL*, pp. 137–140 (2006).

[31] SPSS: Textmining for Clementine http://www.spss.co.jp/software/modeler_ta/.

[32] Sutinen, E.: Approximate Pattern Matching with the q-Gram Family, *Ph.D. thesis, Report A-1998-3, Department of Computer Science, University of Helsinki* (1998).

[33] Takeda, T. and Takasu, T.: UpdateNews: a news clustering and summarization system using efficient text processing, *Proceedings of the 2007 conference on Digital libraries*, pp. 438–439 (2007).

[34] Takeda, T. and Takasu, T.: A Spam Blog Filitering Method Based on Text Copy Detection, *The First IEEE International Conference on the Applications of Digital Information and Web Technologie*, pp. 543–548 (2008).

[35] Thompson, J. D., Higgins, D. G. and Gibson, T. J.: CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice, *Nucleic Acids Research*, Vol. 22, No. 22, pp. 4673–4680 (1994).

[36] Toshida, M. and Haraguchi, M.: Multiple News Articles Summarization based on Event Reference Information, *Proceeding of NTCIR-4* (2004).

[37] Ukkonen, E.: Finding approximate patterns in strings, *Journal of Algorithms*, Vol. 6, pp. 132–137 (1985).

[38] Wang, L., Jiang, T. and Lawler, E.: Approximation algorithms for tree alignment with a given phylogeny, *Algorithmica*, Vol. 16, pp. 302–315 (1996).

[39] WR, P.: Rapid and sensitive sequence comparison with FASTP and FASTA, *Methods in enzymology*, Vol. 183, pp. 63–98 (1990).

[40] Wu, S. and Manber, U.: Fast text searching allowing errors, *Communications of the ACM*, Vol. 35, No. 10, pp. 83–91 (1992).

[41] 池田大介，南野朋之，奥村 学:blog の著者の性別推定，言語処理学会第 12 回年次大会 (2006).

[42] 石田 和成:スパムブログの定量的調査と分離の試み，データベースと Web 情報システムに関するシンポジウム (DBWeb2007) (2007).

[43] 泉　雅貴，三浦孝夫，塩谷 勇:Blog 著者年代推定のためのエントロピによる特徴語抽出，第 19 回データ工学ワークショップ DEWS2008 (2008).

[44] 乾健太郎，藤田 篤:言い換え技術に関する研究動向，自然言語処理，Vol. 11, No. 5, pp. 151–198 (2004).

[45] 奥村学:blog マイニング : インターネット上のトレンド , 意見分析を目指して，人工知能学会誌， Vol. 21, No. 4, pp. 424–429 (2006).

[46] 佐々木拓郎，森 辰則:情報利得比に基づく語の重要度と MMR の統合による複数文書要約，情報処理学会研究報告. 自然言語処理研究会，Vol. 2002, No. 104, pp. 63–70 (2002).

[47] 佐藤有記，宇津呂武仁，福原知宏，河田容英，村上嘉陽，中川裕志，神門 典子:キーワードの時系列特性を利用したスパムブログの収集・類型化・データセット作成，第 19 回データ工学ワークショップ DEWS2008 (2008).

[48] 外間智子，北川 博之:blog における人物に関する 旬な 話題の抽出，第 17 回データ工学ワークショップ DEWS2006 (2006).

[49] 高橋大和，廣嶋伸章，　古瀬蔵，片岡 良治:意見性判定手法の評価と精度向上，言語処理学会第 13 回年次大会 (2007).

[50] 竹田隆治，高須 淳宏:軽量のテキスト処理による部分類似単語列検出手法，電子情報通信学会技術研究報告 人工知能と知識処理，Vol. 107, No. 78, pp. 33–38 (2007).

[51] 成澤和志，山田泰寛，池田 大輔:部分文字列の数え上げによるブログスパムの検出，情報処理学会研究報告. 情報学基礎研究会，Vol. 2006, No. 59, pp. 45–52 (2006).

[52] 難波 英嗣:情報抽出を利用した複数文書要約，日本知能情報ファジィ学会誌，Vol. 18, No. 5, pp. 682–688 (2006).

[53] 平尾　努，奥村　学，福島孝博，難波英嗣，野畑　周，磯崎 秀樹:抜粋による複数文書要約を評価するためのコーパスと評価指標，情報処理学会論文誌，Vol. 48, No. 14, pp. 60–68 (2007).

[54] 廣嶋伸章，山田節夫，古瀬　蔵，片岡 良治:評判検索におけるクエリ依存型の評価極性付与 (意見・評判情報処理)，情報処理学会研究報告. 自然言語処理研究会，Vol. 2006, No. 126, pp. 129–134 (2006).

[55] 丸川雄三，岩山　真，奥村　学，新森 昭宏:ローカルアラインメントを用いたテキスト間の柔軟な対応付け，情報処理学会研究報告 情報学基礎研究会報告，Vol. 2002, No. 87, pp. 23–28 (2002).

[56] 宮部泰成，高村大也，奥村 学:異なる文書中の文間関係の特定，情報処理学会研究報告 自然言語処理研究会， Vol. 105, No. 203, pp. 35–42 (2005).

[57] 渡辺太郎，今村賢治，隅田英一郎，奥乃 博:階層的句アライメントを用いた統計的機械翻訳， Vol. J87-D-II, No. 4, pp. 978–986 (2004).

[58] 渡邊拓也，太田　学，片山　薫，石川 博:分野に依存しない複数文書要約手法の提案，第 15 回データ工学ワークショップ DEWS2004 (2004).

[59] ジャストシステム CB Market Intelligence http://www.justsystems.com/jp/km/cbmi/index.html.

[60] 野村総合研究所 TRUE TELLER http://www.trueteller.net/.

研究業績

□ 学術論文

[総合研究大学入学以後の主著]

- 竹田隆治 高須淳宏: 複製文字列検知に基づいた Splog フィルタリング手法 情報処理学会論文誌　データベース（TOD), Vol. 2, No. 1, pp. 93-103, 2009

□ 国際会議・査読付き講演・レター等 (全て査読有)

[総合研究大学入学以後の主著]

- Takaharu Takeda, Atsuhiro Takasu, Jun Adachi, Kyo Kageura: New Event Detection with Time Subtraction and Co-occurring Words. International Conference on Knowledge Sharing and Collaborative Engineering (KSCE 2006), pp.26-33, 2006.
- Takaharu Takeda, Atsuhiro Takasu. UpdateNews: a news clustering and summarization system using efficient text processing. International Conference on Digital Libraries, Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries. Pages: 438 - 439. 2007
- Takaharu Takeda, Atsuhiro Takasu. News Aggregating System with Automatic Summarization Based on Local Multiple Alignment. The 6th International Conference on Computers and Informatics. Pages: NLP 65-73. 2008
- Takaharu Takeda, Atsuhiro Takasu: A Splog Filtering Method Based on String Copy Detection. IEEE International Conference on the Applications of Digital Information and Web Technologies (ICADIWT), pp.543-548, 2008.

[**上記以外**]

- 竹田隆治 高須淳宏: 軽量のテキスト処理による部分類似単語列検出手法 (「自動化:推論, 発見, 学習, データマイニング」及び一般) 電子情報通信学会技術研究報告. AI, 人工知能と知識処理 Vol.107, No.78(20070524) pp. 33-38 AI2007-7

□ **製品開発**

- 電通バズリサーチ Ver.2.0
- クチコミ@係長
  株式会社ホットリンク