レーベンシュタイン距離の正規化をもちいたログのグループ化による検索時の表示件数の削減

佐藤 健斗[1]　川端 ももの[1]　串田 高幸[1]

概要：ログは，システムの動作状況を記録したデータであり，システム障害の原因特定に使用される．ログの収集から可視化までの過程で，Fluentd，Elasticsearch，Kibana が使用される．Fluentd でログを収集し，Elasticsearch で管理，検索が行われる．Elasticsearch に格納されたデータをもとに，Kibana で可視化する．課題は，管理者が原因調査のためにログを検索する際に，冗長なログによる原因箇所の見逃しにより原因特定に時間がかかることである．提案手法では，比較を行うログのフィールドの定義とレーベンシュタイン距離の正規化をもちいて類似度を算出し，ログをグループ化してブラウザで表示する．基礎実験から算出した類似度をもとに閾値をレーベンシュタイン距離の正規化で 0.76 とし，0.76 を超えた複数のログのグループ化を行う．評価では，提案適用前と提案適用後におけるログの表示件数を比較した．実験では，Elasticsearch に格納された 1998 年サッカーワールドカップのアクセスログを対象とし，クエリを使用してログの表示件数を確認した．提案適用前の検索時の表示件数が 79 件，71 件，87 件である 3 つのクエリを使用して比較を行った．結果は，表示件数が 79 件の場合，30 件のログがグループ化により 14 件になり，表示件数が 63 件となった．その結果，約 20.25%のログが削減された．表示件数が 71 件の場合，23 件のログがグループ化により 10 件になり，表示件数が 58 件となった．その結果，約 18.30%のログが削減された．87 件の場合，16 件のログがグループ化により 7 件になり，表示件数が 78 件となった．その結果，約 10.34%削減された．したがって，類似したログをグループ化することにより検索する際の表示件数を平均で約 16.30%削減することができた．

1. はじめに

背景

ログは，システムやアプリケーションの動作状況や履歴を記録したデータであり，システムの健全性や運用状況を監視するための役割を果たす [1, 2]．また，イベントの発生時刻や発生場所，イベントを説明するテキストメッセージを含み，システムの状態を分析するための重要なデータソースとなっている．このイベントには，正常な動作やエラーの発生が含まれる．管理者はこれらのログをもとに稼働中のシステムの状態を確認し，問題が発生した場合の原因特定に利用する [3].

ログのデータセットの 1 つとして，1998 年のサッカーワールドカップのアクセスログがある [4]．これは，HTML や CSS，画像の静的コンテンツが中心の Web サイトのログである．このアクセスログには，正常にアクセスができたことを示す 200 番台のステータスコードや Web ページが見つからないことを意味する 404 のエラーが含まれている．このログは，8 つのフィールドで構成されている．ワールドカップのアクセスログのフィールドの構成を表 1 に示す.

表 1: ワールドカップのアクセスログのフィールド構成

フィールド	説明
objectID	リクエストされた URL の一意の整数識別子
clientID	リクエストを発行したクライアントの一意の整数識別子
timestamp	リクエストされた時刻
method	クライアントのリクエストに含まれるメソッド
server	どのサーバーがリクエストを処理したか
type	要求されたファイルのタイプ
status	クライアントがリクエストで示された HTTP バージョン応答ステータスコード
size	レスポンスのバイト数

[1] 東京工科大学コンピュータサイエンス学部
〒 192–0982 東京都八王子市片倉町 1404-1

システムには，メンテナンス，セキュリティ，トラフィック分析を目的で作成されたログファイルが存在する [5]．システムやソフトウェアの大規模化，複雑化によりログファイルの件数は，急速に増加する [6]．そのため，効率的に管理することが重要である．収集したログを活用するためには，可視化や検索機能が使用される [7]．こうした目的のために，ログの収集から可視化までの流れにおいて，Fluentd，Elasticsearch，Kibana が使用される．ログの収集から可視化までの流れを図 1 に示す．

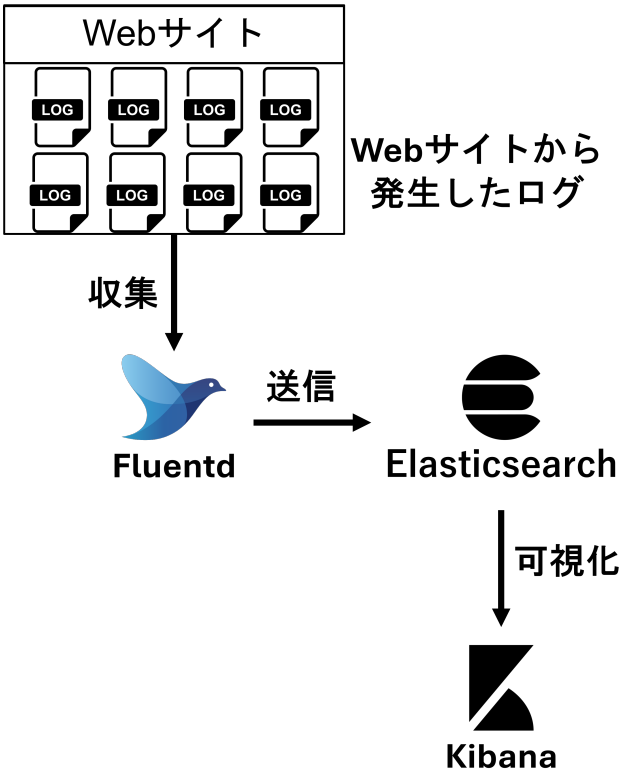

図 1: ログの収集から可視化までの流れ

図 1 では，Web サイト上で発生したログを Fluentd で収集し，Elasticsearch に送信を行っている．その後，Elasticsearch に格納されたデータをもとに，Kibana で可視化を行っている．Fluentd は，データを JSON 形式で構造化する統一アーキテクチャを採用し，データの収集，フィルタリング，バッファリングを行い，複数の出力先や入力元に転送することが可能である [8]．また，プラグインが使用可能なため，多くの外部データソースと接続できる特徴を持つ．Elasticsearch は，大容量のログデータの検索に使用される分散型の全文検索エンジンである [9]．Elasticsearch に収集したデータを視覚的に表示するために Kibana を使用する．Kibana とは，Elasticsearch に格納されたインタラクティブなデータを可視化するための Elasticsearch 用のオープンソースの分析プラットフォームのことである．ログの検索を可視化することによって，管理者は障害の原因箇所を特定することが可能である [10]．また，Apache Lucene クエリ構文を使用して，時間やドキュメントプロパティにもとづいて検索，フィルタリングが可能である [11]．管理者は，システムに障害が発生した際に迅速な原因調査のために Kibana でログを検索する [12]. 表示されたログの中から原因となるログを検索する際に，冗長なデータが含まれるログが存在する．冗長なデータが含まれるログとは，迅速な原因調査の妨げとなるデータで，ログ内のテキスト同士で 1 文字だけ違うログやフィールドの内容が類似しているログである．

課題

課題は，類似しているログによって，ログの見逃しやエラーの原因特定に時間がかかることである．類似しているログの例を図 2 に示す．

図 2: 類似しているログの一例

図 2 は，ワールドカップのアクセスログに含まれる server 以外が類似しているログの例である．A の server は/english/welcoming/news/briefs/autriche_team.jpg である．B の server は/english/welcoming/news/briefs/-tunisia_team.jpg である．A と B のログを比較すると server 以外の要素はすべて一致している．そのため server 以外の要素はエラーの原因特定の際に，冗長なデータが含まれたログとなる．この時，Kibana でログ検索を行うと冗長なデータが含まれたログが表示される．そのため，必要な情報を見つけるのに時間がかかる．

各章の概要

第 2 章では，本稿の関連研究について述べる．第 3 章では，本稿の課題について解決するための提案方式について述べる．第 4 章では，提案した手法の実装について述べる．第 5 章では，提案方式の評価と分析について述べる．第 6 章では，提案方式の議論について述べる．第 7 章では，本

稿のまとめについて述べる．

2. 関連研究

ソフトウェアアプリケーションのロギングインフラストラクチャによって出力されるログデータの量を削減することを目的とした論文がある [13]．この論文は，システム上で疑わしいまたは重要なイベントに関連するログデータの一部に注目することで，オペレーターや自動プロセスが効率的にログを確認できるようにするログ削減手法を提案している．提案手法は，ドメインに依存しないという特徴がある．これは，ドメインの知識を必要とせず，類似性スコアを計算する際に異なるイベントや属性を優先的に処理しないことである．これにより，異なるスキーマ形式に対応できる．しかし，検索クエリにもとづく動的なログの処理に対応できない点に改善の余地がある．

ログパターンの抽出アルゴリズムの改善としてテキストの類似性を用いた手法を提案している論文がある [14]．この論文は，ログ同士の類似性を最長共通部分列 (以後，LCS とする) によって測定し，LCS の長さが一定の割合を超える場合同一パターンとして分類している．これにより，少ない比較回数でパターン分類が可能となり，抽出パターン数と処理時間の両方の改善を行うことができる．しかし，検索クエリにもとづく動的なログの処理に対応できない点に改善の余地がある．

ログデータのクラスタリングを行うアルゴリズムを提案している論文がある [15]. この論文は，テキスト形式のイベントログから，頻繁に発生する行パターンと外れ値のイベントを検出することを目的としている．しかし，ログ全体のパターンをもとにしているため，異なるログの形式やログの種類に対して柔軟な処理ができない点に改善の余地がある．

オンラインサービスシステムのログの特性にもとづく問題特定手法を提案している論文がある [16]．この論文は，LogCluster という手法をもちいて類似したログシーケンスをクラスタにグループ化し，各クラスタから代表的なログシーケンスを抽出している．しかし，オンラインサービスシステムにおいて生成されるログを主な対象としているため，異なるログの種類に対応できない点に改善の余地がある．

3. 提案

基礎実験

基礎実験では，ログの類似度をレーベンシュタイン距離をもちいて算出した．ログ同士の類似度と頻度より，類似しているログとし，グループ化を行う基準値を求めた．レーベンシュタイン距離とは，2 つの文字列間の類似度を計算するための手法である [17]．この距離は，ある文字列を別の文字列に変換するために必要な編集操作の最小回数として定義される [18]．編集操作には，削除，挿入，置換が含まれる．レーベンシュタイン距離における削除の操作の算出方法を式 (1) に示し，挿入の操作を式 (2) に示し，置換の操作を式 (3) にそれぞれ示す．

$$d_{\mathrm{lev}}(i,j) = min(d_{\mathrm{lev}}(i-1,j)+1) \tag{1}$$

$$d_{\mathrm{lev}}(i,j) = min(d_{\mathrm{lev}}(i,j-1)+1) \tag{2}$$

$$d_{\mathrm{lev}}(i,j) = min(d_{\mathrm{lev}}(i-j)+1(a \neq bj)) \tag{3}$$

d は，レーベンシュタイン距離を示している．i と j は，各文字列のインデックスを示している．

基礎実験を行うにあたり，条件による結果の違いを確認するため，2 つの条件を定義した．条件 A は，server の値が最後の/まで一致しているかつ，size と server 以外のフィールドが完全一致しているログとする．条件 A のログの例をコード 1 に示す．size は Log1 では 10457，Log2 では 447 であり，一致していない．server は，Log1 では/english/member/images/member_header，Log2 では/english/member/images/submit.gif であり，最後の「/」までの/english/member/images が一致している．size と server 以外のフィールドは完全一致している．

コード 1: 条件 A の例

```
1 Log1: 38.5.1.0 - - [04/May/1998:22:09:43
      +0000] "GET␣/english/member/images/
      member_header.jpg␣HTTP/1.1" 200 10457
2 Log2: 38.5.1.0 - - [04/May/1998:22:09:43
      +0000] "GET␣/english/member/images/
      submit.gif␣HTTP/1.1" 200 447
```

条件 B は server の値が最後の/までに不一致箇所があるかつ，size と server 以外のフィールドが完全一致しているログとする．条件 B のログの例をコード 2 に示す．size は Log1 では 8389，Log2 では 994 であり，一致していない．server は，Log1 では/images/nav_bg_bottom.jpg，Log2 では/english/images/nav_comp_off.gif であり，最後の「/」以前も不一致である．size と server 以外のフィールドは完全一致している．

コード 2: 条件 B の例

```
1 Log1: 10.5.1.0 - - [04/May/1998:22:06:01
      +0000] "GET␣/images/nav_bg_bottom.jpg␣
      HTTP/1.0" 200 8389
2 Log2: 10.5.1.0 - - [04/May/1998:22:06:01
      +0000] "GET␣/english/images/nav_comp_off
      .gif␣HTTP/1.0" 200 994
```

条件を満たした 1000 組のログでレーベンシュタイン距離の正規化した値を算出し，条件 A と条件 B でそれぞれ度数分布を求めた．条件ごとのレーベンシュタイン距離の正規化の度数分布を図 3 に示す．

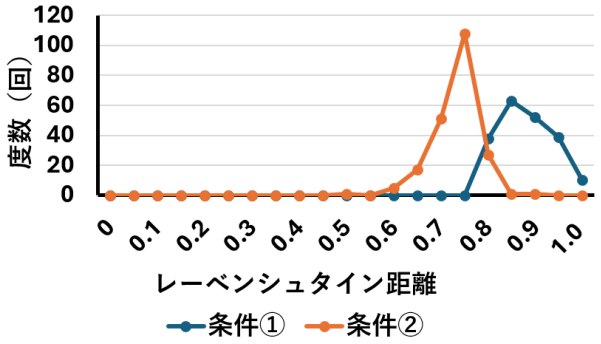

図 3: 条件ごとのレーベンシュタイン距離の正規化の度数分布

図 3 の結果より，条件 A では，レーベンシュタイン距離の値が 0.76 から上がり始め，0.85 で最も多い頻度を示した．条件 B では，レーベンシュタイン距離の値が 0.75 のから下がり始めた．そのため，条件 A を類似度を判定する際の条件とし，類似と判断する閾値を 0.76 以上とする．

提案方式

本稿では，ログの類似度をもちいて検索時の表示件数を削減する手法を提案する．提案ソフトウェアへの入力は，Elasticsearch に格納されたログデータである．出力は，提案ソフトウェアで加工したログデータである．類似度は，レーベンシュタイン距離の正規化をもちいて算出する．2 つのログを比較し，レーベンシュタイン距離の正規化の値が 0.76 以上であった場合グループ化を行う．

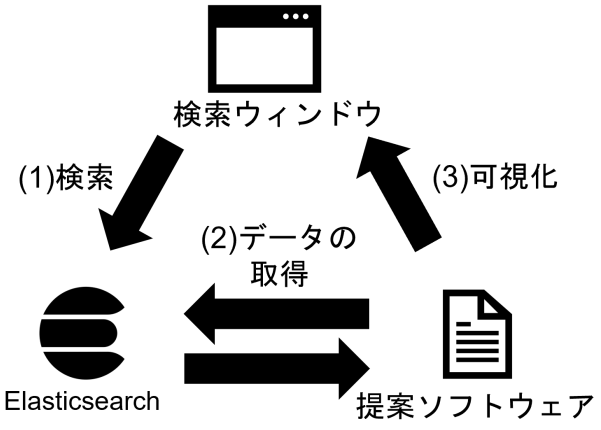

図 4: ログの検索から表示までの流れ

ログの検索から表示までの流れを図 4 に示す．まず，(1) で検索ウィンドウから Elasticsearch に対して検索を行い，Elasticsearch で検索結果のログを出力する．その後，(2) で提案ソフトウェアにより Elasticsearch から検索結果を取得する．提案ソフトウェアを使用し，取得した検索結果の冗長なデータの削除を行い加工したログの検索結果を (3) で検索ウィンドウに表示する．図 5 は，提案ソフトウェアの適用前と適用後のログの表示結果の比較である．提案ソフトウェア適用前は冗長なデータを含むログも表示されているが，提案ソフトウェア適用後は冗長なデータを含むログはグループ化されたため，ログの表示件数が削減されている．そのため，原因調査でログを検索した際に原因特定までの時間を短くできる．

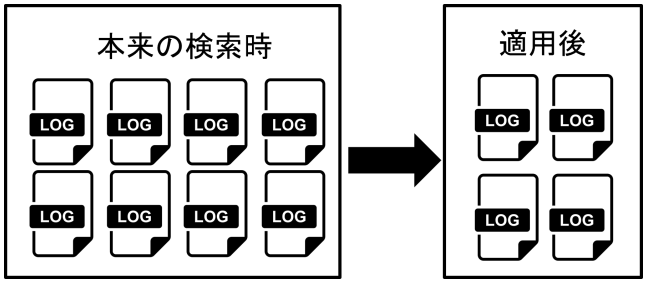

図 5: 提案ソフトウェアをもちいた検索の比較

ユースケース・シナリオ

本稿におけるユースケースとして，Web サイトの運用中に障害が発生し，ユーザーから問い合わせが管理者に送信されている状況を想定する．ログの管理は，Elasticsearch で行う．提案ソフトウェアを使用したユースケース・シナリオを図 6 に示す．

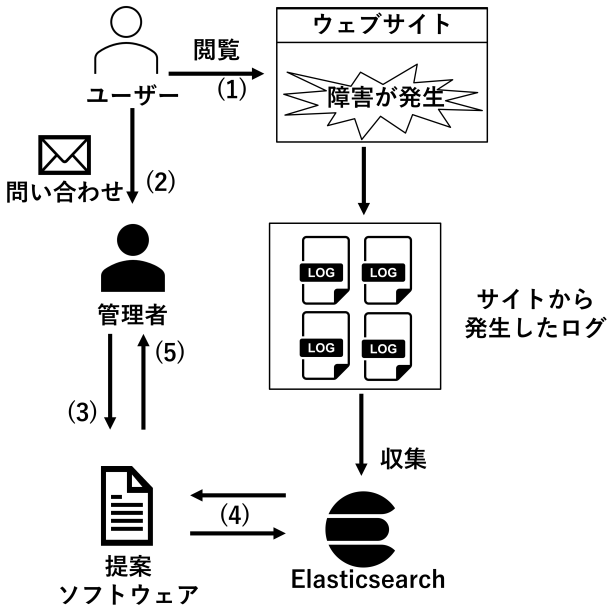

図 6: ユースケース・シナリオ

管理者は，Web サイトの運用保守を行っており，障害が発

生すると原因調査のためにログを検索する．Elasticsearch は Web サイトから生成された，アクセスログを保存している．まず，(1) でユーザーは Web サイトを利用するため，サイトを閲覧する．サイトが閲覧できなかった場合，ユーザーは (2) で管理者に問い合わせを行う．管理者は問い合わせ内容を確認し，原因調査のためクエリを使用してログの検索を行う．その際に，(3) で提案ソフトウェアが実行される．(4) で Elasticsearch からデータを取得し，ログのグループ化を行う．(5) で提案ソフトウェアが適用されたログの検索結果が管理者に表示される．

ログをグループ化することにより，ログを検索する際の表示件数を削減し，原因調査の時間を短縮することができる．

4. 実装

提案手法をもとに，プログラミング言語の Python をもちいて，本稿のソフトウェアを作成した．開発したソフトウェア (grouping.py) の概要を図 7 に示す．提案ソフトウェアには，Web ブラウザ上でログデータを取得するアドレスを指定する機能，クエリを使用した検索機能，検索結果を表示する機能がある．これらの機能は，Python の Web アプリケーションフレームワークの Flask を使用して構築されている．ユーザーが入力フォームにクエリを入力すると，Flask は Elasticsearch の API を利用してクエリを処理し，対応するログデータを取得する．取得されたログデータは，図 7 の流れでログ同士の類似度の計算やグループ化が行われる．

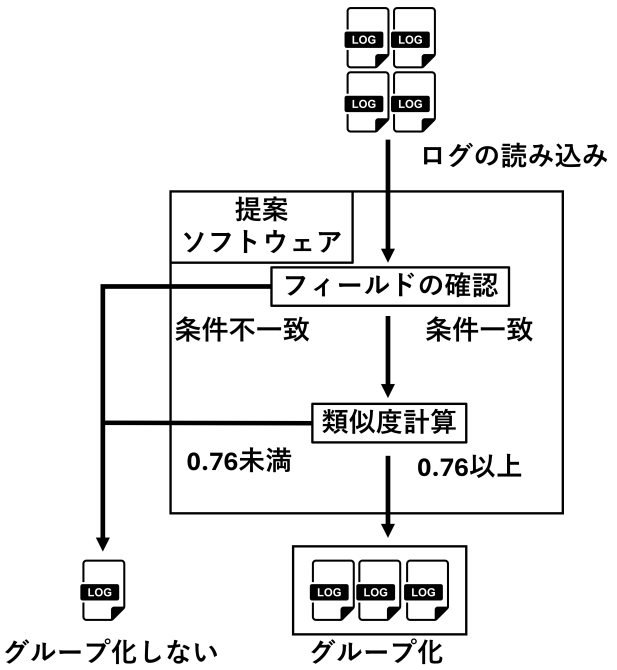

図 7: grouping.py の概要

フィールドの確認では, IP アドレス, timestamp, method, http-verison，status が一致しているか確認を行う．このとき，フィールドの条件に一致していないログのグループ化を行わない．フィールドの確認を行い，条件に一致したログで類似度の計算を行う．類似度はレーベンシュタイン距離を正規化した値とする．このとき，類似度が 0.76 以上の値の場合ログのグループ化を行う．類似度が 0.76 未満の場合は，ログのグループ化を行わない．処理されたログデータは Flask を介してユーザーに Web ブラウザ上で表示される．実装において使用したモジュールと説明を表 2 に示す．

表 2: 使用するモジュール一覧

モジュール名	用途
Flask	Web アプリケーションの構築
requests	Elasticsearch の API への HTTP リクエストを送信，レスポンスの取得，処理
json	JSON 形式のデータ解析，出力
Levenshtein	レーベンシュタイン距離をもちいた類似度の計算
urlparse	URL を解析し，パスやホスト名の構成要素の取得
numpy	計算や多次元配列の操作

表 2 の各モジュールの役割は，Flask は，Web アプリケーションの構築に使用されている．requests は，Elasticsearch の API への HTTP リクエストを送信し，レスポンスを処理するために使用されている．また，API エンドポイントに対してリクエストを送信し，クエリを JSON 形式で渡している．その結果として取得したデータを解析し，ログの情報を抽出して利用している．json は，処理結果を JSON 形式に整形してレスポンスとして渡すために使用されている．Levenshtein は，ログデータ同士の類似度の計算に使用されている．urlparse は，ログデータ内の URL を解析し，パスやディレクトリ部分の取得に使用されている．numpy は，数値データの処理や配列操作を行うために使用されている．

5. 評価実験

評価実験は，提案手法の適用前と適用後でログをクエリを使用して検索をした際の表示件数の比較を行った．

実験環境

実験では，Elasticsearch に保存されている 1998 年のサッカーワールドカップのアクセスログを使用する．使用するログは，1998 年 5 月 5 日の 7 時から 1998 年 5 月 6 日

の 7 時までの期間で 1,664,226 件である．4 台の Virtual Machine(以後，VM とする) を使用する．4 台の VM 全てに vCPU が 3 コア，メモリが 16GB，ストレージが 25GB，OS が Ubuntu22.04 で構成されおり，Kubernetes クラスタが構築されている．Kubernetes クラスタは，マスターノードが 1 台，ワーカーノードが 3 台で構成されている．Kubernetes クラスタ上に Elasticsearch がインストールされている．実験で使用した環境を図 8 に示す．

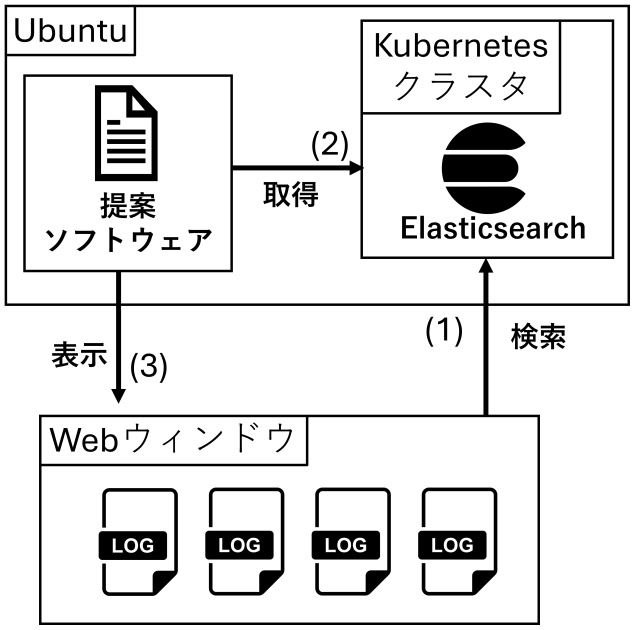

図 8: 実験環境

図 8 では，(1) で Web ウィンドウから Elasticsearch に対して検索を行っている．(2) で提案ソフトウェアで Elasticsearch からログの取得を行っている．(3) で取得したログを提案ソフトウェアで処理し Web ウィンドウにログの表示を行っている．

実験結果と分析

実験で使用したクエリの例をコード 3 に示す．

コード 3: 評価実験に使用するクエリの例

```
{
  "query": {
    "range": {
      "time": {
        "gte": "1998-05-05T07:00:00.000",
        "lte": "1998-05-06T07:00:00.000",
        "format": "yyyy-MM-dd'T'HH:mm:ss.SSS"
      }
    }
  }
}
```

Elasticsearch から 1 度のクエリで取得できるドキュメント数には制限があり，通常は 10,000 件までとなっている．gte,lte の値は，取得するログの開始時刻と終了時刻を示している．クエリを使用して，提案ソフトウェアでログの検索を行った際の表示件数を表 3 に示す．

表 3: ログ検索時の表示件数

	提案適用前の表示件数 (件)	提案適用後の表示件数 (件)
クエリ 1	79	63
クエリ 2	71	58
クエリ 3	87	78

コード 3 で示したクエリで，取得するデータの開始時刻と終了時刻を変更した際の表示件数が，79 件，71 件，87 件である 3 つのクエリを使用して比較を行った．表示件数が 79 件の場合，30 件のログを 14 件にグループ化した．グループは，2 件のログが 1 件にグループ化されたログが 12 組，3 件のログが 1 件にグループ化されたログが 2 組となった．その結果，表示件数が 63 件になり約 20.25%削減された．表示件数が 71 件の場合，23 件のログを 10 件にグループ化した．グループは，2 件のログが 1 件にグループ化されたログが 7 組，3 件のログが 1 件にグループ化されたログが 3 組となった．その結果，表示件数が 58 件になり約 18.30%削減された．87 件の場合，16 件のログを 7 件にグループ化した．グループは，2 件のログが 1 件にグループ化されたログが 5 組，3 件のログが 1 件にグループ化されたログが 2 組となった．その結果，表示件数が 78 件になり約 10.34%削減された．したがって，検索時の表示件数を平均で約 16.30%削減することができた．

6. 議論

本稿の提案手法では，ログ同士の比較を行うフィールドの 1 つとして timestamp を使用している．timestamp は，ログの発生時刻を表している．timestamp も他のフィールドと同様に，テキストとして比較を行っている．そのため，近い時刻でも，レーベンシュタイン距離の値が低いと類似でないと判断されてしまう．例として，timestamp が 11:19 のログに対して，比較するログの timestamp が 11:20 と 12:19 の場合をあげる．timestamp が 11:20 と 12:19 では，11:20 の方が時間差が小さいがレーベンシュタイン距離の操作が 2 回となる．12:19 の場合，時間差が大きいが操作は 1 回となる．そのため，12:19 の方が距離が近くなるため，類似しているログと判断される．この問題に対する改善として，timestamp に関しては，時間差が大きい程，レーベンシュタイン距離の操作の回数に重みづける方法がある．重みづけは，ログの検索時に使用されるクエリの timestamp

に使用される「yyyy-MM-dd'T'HH:mm:ss.SSS」形式をもとにして，1 番下の S を 0 とし，1 桁上がるごとに 1 づつ追加を行う．よって，timestamp でレーベンシュタイン距離の操作を行うと 0 から 16 の重みが操作回数に追加される．この方法を実装することで時間差が大きいログをまとめることを防ぐ. また，レーベンシュタイン距離の操作の重みを追加したため，基礎実験で算出した類似度の分布が変わる．そのため，再度基礎実験を行い，閾値を決定する必要がある．

本稿の提案手法では，閾値を基礎実験より 0.76 とした．しかし，この閾値はデータセットに依存して適切な値を設定する必要がある．現状は，値の選定が固定的であり，他のデータセットに使用できない．この問題に対する改善として，提案ソフトウェアの初期実行時にデータセットを使用してレーベンシュタイン距離の正規化の度数分布を作成し，度数の最低の値を閾値として設定する．これにより，どのデータセットに対しても適切な値を設定することができる．

本稿の提案手法では，定義した条件にもとづいてグループ化するログの閾値を設定した．しかし，異なる条件の場合でも閾値を超えるログが存在するため，条件以外のログもグループ化を行う対象となってしまっている．条件以外のログがグループ化される原因として，server が/french/images/-nav_team_off.gif と/images/backnews.gif の場合，images のように一部一致箇所によりレーベンシュタイン距離の値が高くなることがあげられる．この問題に対する改善として，レーベンシュタイン距離の正規化をもちいた類似度の判断のみではなく，一文字ずつ類似度の判断を追加で行うことで条件以外のログをグループ化されることを防ぐ．類似と判断する基準値は本稿の基礎実験と同様に類似度の度数分布を求め，最低値とする．

7. おわりに

ログは，システムの運用状況を記録するデータであり，システム障害の原因を特定する際に活用される．ログの収集から可視化までの過程で，Fluentd，Elasticsearch，Kibana が使用される．Fluentd をもちいてログを収集し，Elasticsearch で格納されたログを管理，検索し，それを Kibana で可視化する．課題は，冗長なデータを含むログにより原因特定に必要なログが埋もれてしまい，障害の原因特定に時間がかかることである．提案手法では，比較を行うログのフィールドの定義とレーベンシュタイン距離の正規化をもちいて類似度を算出し，ログをグループ化してブラウザで表示する．算出した類似度をもとに，閾値を 0.76 以上と設定し，該当するログをグループ化した．評価では，提案適用前と提案適用後におけるログの表示件数を比較した．実験では，Elasticsearch に格納された 1998 年サッカーワールドカップのアクセスログを対象とし，クエリを使用してログの表示件数を確認した．提案適用前の検索時の表示件数が 79 件，71 件，87 件である 3 つのクエリを使用して比較を行った．結果は，表示件数が 79 件の場合，30 件のログがグループ化により 14 件になり，表示件数が 63 件となった．その結果，約 20.25%のログが削減された．表示件数が 71 件の場合，23 件のログがグループ化により 10 件になり，表示件数が 58 件となった．その結果，約 18.30%のログが削減された．87 件の場合，16 件のログがグループ化により 7 件になり，表示件数が 78 件となった．その結果約 10.34%削減された．したがって，類似したログをグループ化することにより検索する際の表示件数を平均で約 16.30%削減することができた．

謝辞　本稿の執筆にあたりご助言を賜りました, 東京工科大学大学院バイオ・情報メディア研究科コンピュータサイエンス専攻の平尾 真斗さん，東京工科大学コンピュータサイエンス学部の近藤 悠斗さんに御礼申し上げます.

参考文献

[1] Zhu, J., He, S., He, P., Liu, J. and Lyu, M. R.: Loghub: A Large Collection of System Log Datasets for AI-driven Log Analytics, *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*, pp. 355–366 (online), DOI: 10.1109/ISSRE59848.2023.00071 (2023).

[2] Yi, S., Hu, X. and Wu, H.: An automatic reassembly model and algorithm of log file fragments based on graph theory, *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pp. 686–689 (online), DOI: 10.1109/ICSESS.2015.7339150 (2015).

[3] Debnath, B., Solaimani, M., Gulzar, M. A. G., Arora, N., Lumezanu, C., Xu, J., Zong, B., Zhang, H., Jiang, G. and Khan, L.: LogLens: A Real-Time Log Analysis System, *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pp. 1052–1062 (online), DOI: 10.1109/ICDCS.2018.00105 (2018).

[4] Arlitt, M. and Jin, T.: A workload characterization study of the 1998 World Cup Web site, *IEEE Network*, Vol. 14, No. 3, pp. 30–37 (online), DOI: 10.1109/65.844498 (2000).

[5] Sharma, V. and Rawat, S.: Optimizing Forensic Data Availability and Retention of SDN Forensic Logs by Using Bloom Filter, *2023 International Conference on Information Networking (ICOIN)*, pp. 305–311 (online), DOI: 10.1109/ICOIN56518.2023.10048908 (2023).

[6] Zhu, J., He, S., Liu, J., He, P., Xie, Q., Zheng, Z. and Lyu, M. R.: Tools and benchmarks for automated log parsing, *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, IEEE, pp. 121–130 (2019).

[7] Prakash, T., Kakkar, M. and Patel, K.: Geo-identification of web users through logs using ELK stack, *2016 6th International Conference - Cloud System and Big Data Engineering (Confluence)*, pp. 606–610 (online), DOI: 10.1109/CONFLUENCE.2016.7508191 (2016).

[8] Kandan, R., Khalid, M. F., Ismail, B. I., Goortani, E. M., Mydin, M. N. M. and Hoe, O. H.: CLOF: A proposed containerized log management orchestra-

tion framework, *2017 IEEE Conference on Open Systems (ICOS)*, pp. 13–16 (online), DOI: 10.1109/ICOS.2017.8280266 (2017).

[9] Han, L. and Zhu, L.: Design and Implementation of Elasticsearch for Media Data, *2020 International Conference on Computer Engineering and Application (ICCEA)*, pp. 137–140 (online), DOI: 10.1109/ICCEA50009.2020.00036 (2020).

[10] Uday, D. V. and Mamatha, G. S.: An Analysis of Health System Log Files using ELK Stack, *2019 4th International Conference on Recent Trends on Electronics, Information, Communication Technology (RTEICT)*, pp. 891–894 (online), DOI: 10.1109/RTEICT46194.2019.9016706 (2019).

[11] Bajer, M.: Building an IoT Data Hub with Elasticsearch, Logstash and Kibana, *2017 5th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, pp. 63–68 (online), DOI: 10.1109/FiCloudW.2017.101 (2017).

[12] Zamfir, V.-A., Carabas, M., Carabas, C. and Tapus, N.: Systems Monitoring and Big Data Analysis Using the Elasticsearch System, *2019 22nd International Conference on Control Systems and Computer Science (CSCS)*, pp. 188–193 (online), DOI: 10.1109/CSCS.2019.00039 (2019).

[13] Kalamatianos, T., Kontogiannis, K. and Matthews, P.: Domain Independent Event Analysis for Log Data Reduction, *2012 IEEE 36th Annual Computer Software and Applications Conference*, pp. 225–232 (online), DOI: 10.1109/COMPSAC.2012.33 (2012).

[14] Zhao, Y., Wang, X., Xiao, H. and Chi, X.: Improvement of the Log Pattern Extracting Algorithm Using Text Similarity, *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 507–514 (online), DOI: 10.1109/IPDPSW.2018.00087 (2018).

[15] Vaarandi, R. and Pihelgas, M.: LogCluster - A data clustering and pattern mining algorithm for event logs, *2015 11th International Conference on Network and Service Management (CNSM)*, pp. 1–7 (online), DOI: 10.1109/CNSM.2015.7367331 (2015).

[16] Lin, Q., Zhang, H., Lou, J.-G., Zhang, Y. and Chen, X.: Log Clustering Based Problem Identification for Online Service Systems, *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*, pp. 102–111 (2016).

[17] Soyusiawaty, D. and Rahmawanto, F.: Similarity Detector on the Student Assignment Document Using Levenshtein Distance Method, *2018 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, pp. 656–661 (online), DOI: 10.1109/ISRITI.2018.8864339 (2018).

[18] Sugiarto, Diyasa, I. G. S. M. and Diana, I. N.: Levenshtein Distance Algorithm Analysis on Enrollment and Disposition of Letters Application, *2020 6th Information Technology International Seminar (ITIS)*, pp. 198–202 (online), DOI: 10.1109/ITIS50118.2020.9321030 (2020).

