

ビッグデータ処理技術を用いた Wikipedia マイニング

プロジェクトマネジメントコース

ソフトウェア開発管理グループ

矢吹研究室

1242005

石井康之

謝辞

本研究を進めるにあたり，矢吹研究室矢吹太郎准教授には，多くの時間をご指導にさいて頂きました．また矢吹研究室の皆様には，多くの知識や示唆を頂きました．協力していただいた皆様に感謝の気持ちと御礼を申し上げます．

目次

第 1 章	序論	7
第 2 章	背景	9
2.1	研究背景	9
2.2	Wikipedia とは	9
2.3	ビッグデータとは	15
第 3 章	目的	21
第 4 章	手法	23
4.1	VirtualBox とは	23
4.2	VirtualBox のインストール	24
4.3	用語	30
4.4	Ubuntu とは	31
4.5	Ubuntu のインストール	31
4.6	端末	33
4.7	MySQL	34
4.8	Wikipedia の編集履歴データの取得	35
4.9	XML ファイル	37
4.10	XML パーサ	37
第 5 章	結果	39
5.1	Wikipedia の履歴の調査	39
第 6 章	考察	51
第 7 章	結論	53
	参考文献	55

第 1 章

序論

当研究はビッグデータ処理技術を用いた Wikipedia マイニングを行う。

Wikipedia とは、非営利の Wikimedia 財団がインターネット上で運営する、無料のオンライン百科事典プロジェクトである。記事の内容はボランティアの人々の協力によって、信頼のおける品質が保たれている。

ビッグデータとは、市販されているツールや従来のデータ処理で行うことが困難なほど巨大なデータ集合の集積物のことである。

先行研究では、Google BigQuery という、Google Cloud Platform が提供するビッグデータ解析サービスを扱った。Google BigQuery のサイトの中に、サンプルデータとして、Wikipedia の編集履歴データを提供していたので、それを用いて研究を行った。しかし、提供されていたデータは英語版だけのものであり、多言語の研究を行うことができなかったため、別の研究方法を検討する必要がある。

そこで当研究は、多言語版 Wikipedia もデータ解析できる技術を得るために、ローカルで研究が行える環境を用意し、データマイニングを行う。

Wikipedia の全編集データをマイニングすることによって、Wikipedia の品質が保たれている理由を見つけ出す。

第 2 章

背景

2.1 研究背景

Wikipedia は、多くのボランティアにより、始まってから 10 年足らずの間に、大きな成長を見せたオンライン百科事典プロジェクトである。総記事数の文字数は 10 億文字を超え、ブリタニカ国際大百科事典とエンカルタ総合大百科の合計と比較しても上回る。Wikipedia は、さまざまな言語が参加しているグローバルなプロジェクトでもある。2015 年 9 月までには、291 個もの言語が参加している。

このオープンなプロジェクトの百科事典は、制限無く誰でも自由に使用でき編集することもできる。

誰でも自由に編集できるからこそ、ボランティアの人々は気軽に参加でき、特定の企業や個人のお金を稼ぐのに力を貸していると感じることなく、時間と労力を注ぐことができる。

記事の内容はボランティアの人々の協力によって、信頼のおける品質が保たれている。しかし、中には協力的では無く、悪意のある編集をするものがある。悪意のある編集者はその記事の内容とは関係ないことを書き込んだり、記事の破壊行為を繰り返している。Wikipedia では、悪意のある編集をする人とわかっていても規制などをしたりはしない。記事は完成・確定されることはなく、新しい情報にいつでも改変することができる。

本研究では、Wikipedia の全編集データをマイニングすることによって、Wikipedia の品質が保たれている理由を見つけ出す。

2.2 Wikipedia とは

フリー・ライセンスの百科事典である。フリーには 2 つの意味がある。無料という意味と、自由という意味だ。Wikipedia のフリーは後者の自由という意味であり、四つの自由が与えられている。著作物を複製する自由、改変する自由、再頒布する自由、そして、改変版を再頒布する自由だ。そして、営利目的に使っても、非営利に使ってもかまわない。というものがある。Wikipedia がフリーの百科事典であるというのは、無料でアクセスできるということではなくて、自由に複製、改変、利用してかまわないということである。

Wikipedia という名前は、ウェブブラウザ上でウェブページを編集することができる Wiki というシステムを使用した百科事典であることに由来する造語である。設立者の 1 人であるラリー・サンガーにより命名された。

Wikipedia は 2015 年 9 月までには、291 個もの言語が参加している。この百科事典は多くの言語のボランティアたちによって書かれたグローバルなプロジェクトでもある。[1]

2.2.1 記事の編集の仕方

一部の保護されているページを除いて、全てのページには「編集」と書かれたリンクがあり、このリンクを使って、あなたが閲覧しているページを編集することができます。編集ができることはウィキペディアの大きな特徴で、この機能を使って、あなたが記事を修正したり、記事に加筆することができるのです。記事に情

報を加筆する時には、情報の出典を明記してください。出典が不明な記述は、除去の対象となります。[2]

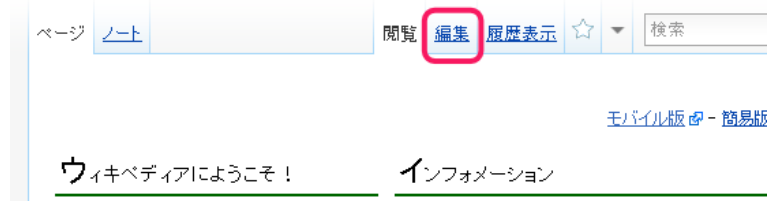


図 2.1 図の挿入例

これから常に使ってほしい大切な機能が「プレビューを表示」ボタンです。サンドボックスでなにか編集をして、それから「以上の記述を完全に理解し同意した上で投稿する」ボタンではなく、「プレビューを表示」ボタンを押してみましょう。そうすると、あなたがページに加えた変更の結果を、実際に保存する前に確認することができます。間違いは誰にでもあります。この機能は、間違いがないか自分で確認するためのものです。また、「プレビューを表示」ボタンを使えば、試しにページの体裁や表現をいろいろと変えてみても、ページの変更の記録にいちいち記録されずにすみますし、他にもいろいろと利点があるのです。でも、プレビューをした後、最後には保存するのを忘れないでください。[2]

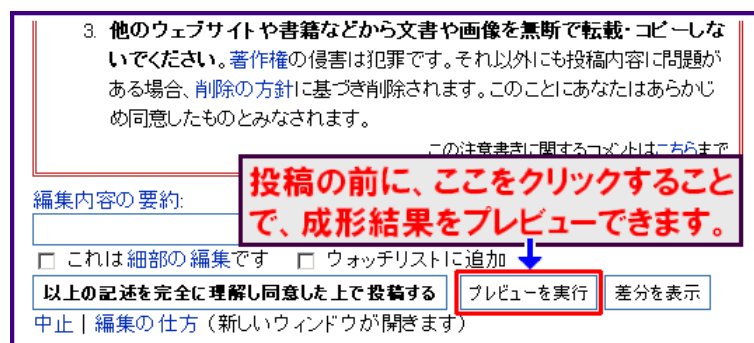


図 2.2 図の挿入例

「以上の記述を完全に理解し同意した上で投稿する」ボタンを押す前に、あなたが行った編集の説明を、編集用のテキストボックスと保存ボタンの間にある要約欄に書き込むようにしましょう。ウィキペディアでは、ここに編集の説明を書き込むことが大切なエチケットと考えられています。ただ単に誤字を直したような時には「誤字修正」と書けば充分です。文章の意味に影響を及ぼさないような、小さな修正のときには、要約欄の下にある「これは細部の編集です（説明）」のチェックボックスにチェックをいれておいてください（この機能はログイン時にのみ有効です）。[2]



図 2.3 図の挿入例

2.2.2 Wikipedia の編集履歴データ

データ Creative Commons Attribution-ShareAlike 3.0 Unported License (CC-BY-SA) および GNU Free Documentation License (GFDL) の下にライセンスされており（Wikipedia:著作権および利用規約を参照）、再配布や再利用のためにデータベース・データの提供が行われています。データの生成は不定期に行われている。

Wikipedia ではクローリング行為のデータダウンロードは禁止されている．強引なクローリングは，Wikipedia が劇的に遅くなる原因となってしまいますためである．データベースから自動的にデータ収集している行為が発券された場合、システムの管理者から自身のサイトから Wikipedia のアクセスを禁止されてしまう措置が起こってしまうこともある．また，ウィキペディア財団が法的措置を検討する場合もあるので，注意が必要．

Index of /jawiki/

../		
20150118/	21-Jan-2015 04:39	-
20150227/	24-Feb-2015 17:51	-
20150313/	16-Mar-2015 14:37	-
20150409/	06-Apr-2015 06:19	-
20150422/	25-Apr-2015 13:52	-
20150512/	15-May-2015 08:17	-
20150602/	16-Jun-2015 01:34	-
20150703/	08-Jul-2015 14:44	-
20150805/	13-Aug-2015 21:29	-
20150826/	29-Aug-2015 04:03	-
20150901/	10-Sep-2015 23:02	-
latest/	10-Sep-2015 23:02	-

図 2.4 図の挿入例

ここに日本語版 Wikipedia の履歴データが記録されている．

URL は <https://dumps.wikimedia.org/jawiki/>

他の言語もこのような形式で履歴データが残されている．他の言語のデータを取得したい場合は URL の <https://dumps.wikimedia.org/> wiki/ の部分を変更すればよい．言語は英語のスペルで頭文字 2 文字でよい．

例：英語の場合はスペルは English なので，<https://dumps.wikimedia.org/enwiki/> とすればよい．

Index of /jawiki/latest/

../		
jawiki-latest-abstract.xml	08-Sep-2015 18:41	1823775581
jawiki-latest-abstract.xml-rss.xml	08-Sep-2015 18:41	751
jawiki-latest-abstract1.xml	08-Sep-2015 18:36	643971794
jawiki-latest-abstract1.xml-rss.xml	08-Sep-2015 18:40	754
jawiki-latest-abstract2.xml	08-Sep-2015 18:24	416000876
jawiki-latest-abstract2.xml-rss.xml	08-Sep-2015 18:40	754
jawiki-latest-abstract3.xml	08-Sep-2015 18:40	397852002
jawiki-latest-abstract3.xml-rss.xml	08-Sep-2015 18:40	754
jawiki-latest-abstract4.xml	08-Sep-2015 18:24	365950954
jawiki-latest-abstract4.xml-rss.xml	08-Sep-2015 18:40	754
jawiki-latest-all-titles-in-ns0.gz	08-Sep-2015 16:33	9871154
jawiki-latest-all-titles-in-ns0.gz-rss.xml	08-Sep-2015 16:33	775
jawiki-latest-all-titles-in-ns0.gz-rss.xml	08-Sep-2015 16:33	17646032
jawiki-latest-all-titles-in-ns0.gz-rss.xml	08-Sep-2015 16:33	754
jawiki-latest-category.sql.gz	02-Sep-2015 13:42	3115854
jawiki-latest-category.sql.gz-rss.xml	08-Sep-2015 16:33	760
jawiki-latest-categorylinks.sql.gz	02-Sep-2015 13:35	139172740
jawiki-latest-categorylinks.sql.gz-rss.xml	08-Sep-2015 16:33	775
jawiki-latest-externalinks.sql.gz	02-Sep-2015 13:41	186863400
jawiki-latest-externalinks.sql.gz-rss.xml	08-Sep-2015 16:33	775
jawiki-latest-geo-tags.sql.gz	02-Sep-2015 13:43	1324387
jawiki-latest-geo-tags.sql.gz-rss.xml	08-Sep-2015 16:33	760
jawiki-latest-image.sql.gz	02-Sep-2015 13:08	11887582
jawiki-latest-image.sql.gz-rss.xml	08-Sep-2015 16:33	751
jawiki-latest-image.sql.gz-rss.xml	08-Sep-2015 16:33	27830530
jawiki-latest-imagelinks.sql.gz	08-Sep-2015 16:33	766
jawiki-latest-imagelinks.sql.gz-rss.xml	02-Sep-2015 13:42	732
jawiki-latest-interwiki.sql.gz	08-Sep-2015 16:33	763
jawiki-latest-interwiki.sql.gz-rss.xml	02-Sep-2015 13:43	21926248
jawiki-latest-lwlinks.sql.gz	08-Sep-2015 16:33	757
jawiki-latest-lwlinks.sql.gz-rss.xml	02-Sep-2015 13:42	100477638
jawiki-latest-langlinks.sql.gz	08-Sep-2015 16:33	763
jawiki-latest-langlinks.sql.gz-rss.xml	10-Sep-2015 23:02	4420
jawiki-latest-mdsums.txt	02-Sep-2015 13:43	108858170
jawiki-latest-page.sql.gz	08-Sep-2015 16:33	748
jawiki-latest-page.props.sql.gz	02-Sep-2015 13:43	31465947
jawiki-latest-page.props.sql.gz-rss.xml	08-Sep-2015 16:33	766
jawiki-latest-page.restrictions.sql.gz	02-Sep-2015 13:43	82481
jawiki-latest-page.restrictions.sql.gz-rss.xml	08-Sep-2015 16:33	787
jawiki-latest-pagelinks.sql.gz	02-Sep-2015 13:33	523757143
jawiki-latest-pagelinks.sql.gz-rss.xml	08-Sep-2015 16:33	763
jawiki-latest-pages-articles-multistream-index.txt	03-Sep-2015 08:46	19678416

図 2.5 図の挿入例

どれか開くと上記のような画面になる．

ウィキページのデータは SQL のテーブルではなく、XML で提供されている。XML ファイルの文字エン

コーディングは UTF-8 である。非常にファイルサイズが大きいため、通常のエディタやブラウザで、解凍してはいけない。

データの詳細は下記のとおり

- pages-articles.xml.bz2 - ノートページ、利用者ページを除く最新版のダンプ
- pages-meta-current.xml.bz2 - 全ページの最新版のダンプ
- pages-meta-history.xml.7z - 全ページの全ての版のダンプ
- all-titles-in-ns0.gz - 全項目のページ名一覧 (標準名前空間)

2.2.3 Wikipedia の編集回数の多いページの一覧

期間: 2014-07-01 2014-07-31 のランキング。

順位 [↗]	ページ [↗]	編集回数 [↗]	総編集回数 [↗]
1 [↗]	利用者:タベストリー/sandbox [↗]	651 [↗]	917 [↗]
2 [↗]	Wikipedia:管理者伝言板/投稿ブロック/ソックパペット [↗]	379 [↗]	3098 [↗]
3 [↗]	利用者:ワナー成増/sandbox [↗]	376 [↗]	2475 [↗]
4 [↗]	Wikipedia:管理者伝言板/投稿ブロック/history20140727 [↗]	368 [↗]	4090 [↗]
5 [↗]	Wikipedia:メインページ新着投票所/新しい項目候補 [↗]	328 [↗]	10594 [↗]
5 [↗]	ハピネスチャージプリキュア! [↗]	328 [↗]	1878 [↗]
7 [↗]	FNS27 時間テレビ (2014 年) [↗]	306 [↗]	306 [↗]
8 [↗]	Wikipedia:改名提案/history20140727 [↗]	283 [↗]	6760 [↗]
9 [↗]	利用者:Tribot/log [↗]	272 [↗]	1720 [↗]
9 [↗]	利用者:ワナー成増/下書き [↗]	272 [↗]	363 [↗]
11 [↗]	2014 年のテレビ (日本) [↗]	237 [↗]	2321 [↗]
12 [↗]	インテリビレッジの座敷童 [↗]	229 [↗]	233 [↗]
13 [↗]	洪門 [↗]	209 [↗]	243 [↗]
14 [↗]	義経=ジンギスカン説 [↗]	200 [↗]	716 [↗]
15 [↗]	妖怪ウォッチ [↗]	198 [↗]	618 [↗]
16 [↗]	スカッとゴルフ パンヤ [↗]	197 [↗]	1514 [↗]
17 [↗]	笑福亭べ瓶 [↗]	192 [↗]	394 [↗]
18 [↗]	博士と助手〜細かすぎて伝わらないモノマネ選手権〜 [↗]	178 [↗]	1503 [↗]
19 [↗]	マレーシア航空 17 便 [↗]	168 [↗]	168 [↗]
20 [↗]	小保方晴子 [↗]	161 [↗]	862 [↗]
20 [↗]	Wikipedia:リダイレクトの削除依頼/2014 年 7 月 [↗]	161 [↗]	161 [↗]
22 [↗]	大相撲力士一覧 [↗]	151 [↗]	1499 [↗]
22 [↗]	花子とアン [↗]	151 [↗]	1147 [↗]
22 [↗]	利用者:チンドレ・マンドレ/sandbox [↗]	151 [↗]	516 [↗]
22 [↗]	ノート:集団的自衛権 [↗]	151 [↗]	280 [↗]
26 [↗]	ALDNOAH.ZERO [↗]	149 [↗]	190 [↗]
27 [↗]	ノート:野々村竜太郎 [↗]	143 [↗]	143 [↗]
28 [↗]	赤穂市 [↗]	142 [↗]	707 [↗]
29 [↗]	STAP 研究と騒動の経過 [↗]	140 [↗]	180 [↗]
30 [↗]	Wikipedia:コメント依頼/みしまるもも 20140528 [↗]	139 [↗]	209 [↗]
31 [↗]	GENEZ [↗]	136 [↗]	259 [↗]

32	ジェンパクト・ヘッドストロング・ビジネスコンサルティング	134	134
33	Wikipedia:保護依頼:history20140727	129	4587
34	静岡市	122	3252
35	利用者:ワーナー成増	119	419
36	利用者:ワーナー成増/下書き 2	117	176
37	計報 2014 年	116	877
38	利用者:Gowithitjam/sandbox	115	115
39	仮面ライダー鎧武/ガイム	113	2793
39	帝京大学	113	2743
41	2014 FIFA ワールドカップ	112	607
42	ドラえもん (1979 年のテレビアニメ) の帯番組時エピソード一覧	111	150
43	パワーパフガールズ	109	3655
43	利用者:Psychotic Blue/下書き 2	109	896
43	利用者:南北円上王	109	285
43	Wikipedia- ノート:管理者への立候補	109	265
47	烈車戦隊トッキュウジャー	108	958
48	Wikipedia:コメント依頼/history20140727	106	3119
48	利用者:Tamrono157/サンドボックス	106	144
50	パナソニックショップ	104	851
50	2014 FIFA ワールドカップ・決勝トーナメント	104	140
52	利用者:会話:Enyokoyama/sandbox	101	251
53	金田一少年の事件簿 (テレビドラマ)	100	1208
53	東海中学校・高等学校	100	1028
53	刺激惹規性多能性獲得細胞	100	766
56	さばげぶっ!	99	171
56	ドラえもん (1979 年のテレビアニメ) のエピソード一覧 (2001 年 - 2005 年)	99	119
58	Wikipedia統合提案/history20140727	97	4536
59	Wikipedia削除の復帰以来	96	1833
59	HERO (テレビドラマ)	96	841
59	RAIL WARS! -日本国鉄鉄道公安隊-	96	199
62	利用者:舍利弗/アンコール・ワット	95	95
63	SASUKE	94	4100

64	Wikipedia議論が盛んなノート	93	646
65	2014年の日本競馬	90	1063
66	うえのやまさおり	88	88
67	金田一少年の事件簿の犯罪者	86	907
67	利用者:Ajikoube-828/sandbox	86	308
69	国際プロレス	85	663
69	GODZILLA ゴジラ	85	525
69	実況パワフルプロ野球 2013	85	488
69	利用者:会話:南北円上王	85	145
73	2014年のオールスター（日本プロ野球）	83	83
74	ハマトラ（アニメ）	81	340
75	利用者:Quark Logo/sandbox3 文禄・慶長の役	80	91
75	星亮一	80	80
75	俺の屍を超えて行け 2	80	217
75	入江仁之	80	215
79	家族狩り	79	132
79	山下達郎	79	2486
79	ヘイトスピーチ	79	686
79	ノート:ゼーロン	79	79
79	ノート:橋本環奈	79	79
84	森川智之	78	2599
84	白雪姫	78	416
84	バイナリーオプション	78	89
87	Wikipedia:分割提案	77	1645
88	橋本環奈	76	161
88	利用者:やまさきなつこ/sandbox	76	128
88	牛丸謙吾	76	76
91	DDT プロレスリング	75	1596
91	利用者:K s/sandbox	75	1375
91	利用者:Iso10970/sandbox	75	205
94	ウルトラマンギンガ S	74	112
95	ガールズ&パンツァー	73	1271
95	チェルシーFC	73	1176
95	まじもじるも	73	140
98	Wikipedia:Bot 作業依頼	72	2161

98	札幌競馬場	72	688
100	浪曲	71	731
100	集団的自衛権	71	131
100	利用者:会話:B side of the moon	71	112

2.2.4 日本語版 Wikipedia について

日本語版の Wikipedia は、英語版 Wikipedia と比べると非常にユニークに見える。まず編集者として行動するときに、日本のウィキペディア編集者は、匿名が多い。これは元々ネットを使うことにおいて匿名でいるが多い日本のインターネット文化が大きな要因である。日本のオンライン活動に大きな影響を与えているサイト

に、「2ちゃんねる」というものがある。2ちゃんねるは、匿名の投稿で有名なサイトである。日本のウィキペディア編集者あえてユーザー登録をしない理由のひとつとして、自分の身元を明かさない2ちゃんねるの「完全な匿名性」の普及が上げられることが多いといえる。

また、日本語版ウィキペディアの編集者は、英語版のような激しい編集合戦はあまり行われないう傾向がある。礼儀正しい日本の文化では、長きにわたって繰り上げられる卑劣な論争は、一般的に社会には受け入れられないからだ。日本のユーザーは欧米と比べるとおとなしい。公開された既存の記事を思い切って変更などもしない。代わりに、他の場所やノート・ページで別のバージョンを考えることが多い。

日本語版ウィキペディアの欠点は、登録ユーザーが少ないことだ。先ほどでも挙げたとおり、匿名性の普及が多いことが要因といえる。登録ユーザーの数が少ないので、ウィキペディア・ユーザーの国際コミュニティや、全プロジェクトを取りまとめる非営利のウィキペディア財団への参加が少ないという問題がある。

日本語版ウィキペディアの規模はトップクラスであるが、2005年にフランクフルトで開催された第一回のウィキペディア会議のウィキマニアでは、日本人の登録参加者は二名だけだった。日本語版よりも小規模なフランス、ポーランド、オランダや、さらに小規模な中国でさえ、日本語版ウィキペディアよりも多くの代表者が参加していた。このことから、日本語版ウィキペディアの欠点はやはり登録しているユーザーが少ないということがあげられる。

2.3 ビッグデータとは

パソコン、スマートフォンが普及し「ビッグデータ」という言葉が流行することからもわかるように、私たちは膨大な情報を日々生み出しながら生活している。GoogleやYahoo!に寄せられる大量の検索クエリや、Twitter、FacebookなどのSNSに投稿される文章や画像、動画、スマートフォンを利用するサービスなどで収集される位置情報データ、防犯カメラで記録される人間の表情や動きのデータなどの膨大な量のデータを指す。

ビッグデータとは、一般的にペタ(1,000兆)、バイト級のデータ量といわれている。このような数値的定義もあるが、ペタバイト以下であればビッグデータではないという訳でもなく、本質的には、「従来の手段では管理しきれない規模のデータ」を指す。

10^n	接頭辞	記号	漢数字表記	十進数表記	分類
10^{21}	ゼタ (zetta)	Z	十垓	1,000,000,000,000,000,000,000	ビッグデータ
10^{18}	エクサ (exa)	E	百京	1,000,000,000,000,000,000	ビッグデータ
10^{15}	ペタ (peta)	P	千兆	1,000,000,000,000,000	ビッグデータ
10^{12}	テラ (tera)	T	一兆	1,000,000,000,000	
10^9	ギガ (giga)	G	十億	1,000,000,000	
10^6	メガ (mega)	M	百万	1,000,000	
10^3	キロ (kilo)	K	千	1,000	

4V による BigData の定義

IBMによるBigDataの定義で4Vというものがある。4Vとは、容量 (Volume)、種類 (Variety)、頻度・スピード (Velocity)、正確さ (Veracity) から構成されている。

容量 (Volume)

ビッグデータの特徴である容量の巨大さを指す。企業内外にはデータが溢れており、数テラバイトから数ペタバイトにもおよぶ。またデータが増大することによる計算量も非常に膨大となる。

種類 (Variety)

ビッグデータは企業システムで通常扱っているような顧客情報や販売データ、経理データ、在庫データなど

の構造化データであるとは限らない。テキスト、音声、ビデオ、ページ遷移、ログファイルなどのさまざまな種類の非構造化データも存在する。

頻度・スピード (Velocity)

今この瞬間にも、ものすごい頻度で RFID などの IC タグやセンサーなどからデータが生成されている。昨今の変化の著しい市場環境では、これらのデータによりリアルタイムに対応したものを求められている。

正確さ (Veracity)

データの矛盾、曖昧さによる不確実性、近似値を積み重ねた不正確などを排除して、本当に信頼できるデータが意思決定には重要である。

以上が IBM による 4V の定義であるが、容量 (Volume) については最初に記述したように、必ずしもペタバイト以上でなければならないとは考えていない。

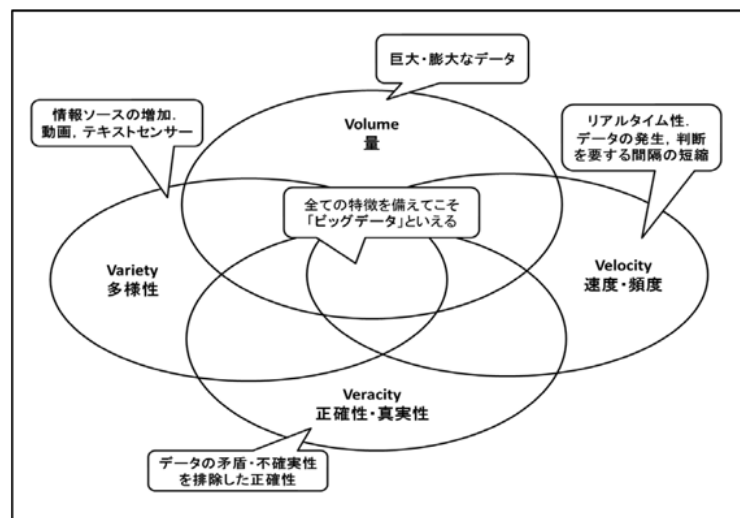


図 2.6 ビッグデータの 4V

3V でのビッグデータの定義

3V で表す場合は、容量 (Volume)、種類 (Variety)、頻度・スピード (Velocity) の 3 つになり、正確さ (Veracity) は含まれない。確かにビッグデータには正確でないデータが混在することもあり、例えば Twitter などの SNS データには、冗談やデマ情報の書き込みなども混じっており、センサーなどでも故障によるノイズが混じることもあります。データ量が少ない場合には、外れ値として手作業で除去することも可能だがいわゆるビッグデータと言われている大量のデータの場合は、手作業によるデマ情報やノイズなどの除去はほとんどふかのうである。

しかし、ビッグデータで収集するデータは殆どが生データであるという特徴がある。正確さをどう定義するにもよるが、センサーからの入力データなどは生データそのものだが、SNS からの入力データなども生データであり、その意味では正確なデータといえる。つまり、編集などで手が加わっていないデータであり、またそこで発せられるメッセージはその人が、その人の環境により制約などを感じるデータではないからだ。これは、勤務する企業・組織内で作成する報告書など対比して考えるとわかりやすいだろう。

ビッグデータ処理のパターン

下記の表に、3 つの処理パターンの特性を簡潔にまとめたものを記述する。

	バッチ処理	インタラクティブクエリー処理	ストリームデータ処理
実行タイミング	ユーザー指定と定期的実行	ユーザー指定と定期的実行	常時連続実行
処理単位	蓄積データをバッチで一括処理	蓄積データをバッチで一括処理	少数のフローデータ処理
実行時間	分～時間	秒～分	ミリ秒～秒
処理モデル	MapReduce	クエリ・OLTP	ストリーム処理

ビッグデータの処理パターンには、「バッチ処理」、「インタラクティブクエリー処理」、「ストリームデータ処理」の3種類のパターンがある。

バッチ処理では蓄積データをバッチで一括処理だが、これは Google 検索用に開発された MapReduce 処理を利用した Hadoop が代表的である。しかし、Hadoop はビッグデータ処理用として開発されたものではないので、処理結果作成に時間がかかるという欠点がある。

インタラクティブクエリー処理は、蓄積された大容量データをオンラインクエリなど使用して一括解析処理するものである。インタラクティブクエリー処理では蓄積されたビッグデータを数秒から数分で実行する。

ストリームデータ処理は大量発生する実世界データを逐次に時系列処理する技術である。データ発生時にあらかじめ登録したシナリオにしたがって集計・分析に必要なデータを抽出し、データ処理を行う。このように逐次時系列でデータ処理できることから、最新の情報、その中での特異な値の発生などに対してリアルタイムに対応するシステムを構築できることが特徴で IoT への応用に最適な処理方法といえる。[3]

バッチ処理

最初に MapReduce で代表される、バッチ処理の特性を見る。

数十年前のメインフレームは、主記憶が数百キロバイト、価格は数千万程度だった。これを現在の PC（主記憶数ギガバイト、価格は 10 万円程度）と比べた場合、価格性能比では約 100 万倍にもなる。また、CPU 処理スピードと、ネットワークの帯域幅についてはそれぞれ、主記憶と類似の性能向上を遂げてきている。

このようなことは、コンピューター関係以外の業種ではまったく例を見ない群を抜く性能向上である。この急激なプラットフォームの真価がクラウドコンピューティングやそのうえで実行されるビッグデータ処理などを可能にしている。[3]

ただし、これはクラウドなどに限ったことではない。IT の世界ではこれまでも短いタイムスパンで新しいテクノロジー・ブレークスルーやビジネスモデルが出現してきており、これはプラットフォームやネットワークの新派に依存している部分が多くある。言葉を変えれば、これらの新しい発想はその時点でのプラットフォーム性能で初めて成り立つものであり、これをわずかも前の世代に思いついたとしても、実現不可能である場合が多い。

このようにクラウドなどの先端 IT システムは、現在のそしてこれからも進化を続けるはずのプラットフォームやネットワークに依存したものである。[3]

インタラクティブクエリー処理

インタラクティブクエリー処理は、BigQuery による説明を記述する。Google がリリースするソフトウェアツールには、もともと Google が社内使用の目的で開発していたものも多く、BigQuery もそれに当てはまる。Google も当初は社内使用でも MapReduce を使用してビッグデータ処理を行っていたが、バッチ処理による結果生成の遅延や処理を行うための準備の煩雑さなどから、それに代わるツールとして開発されたのが BigQuery である。BigQuery はデータの入力または JSON フォーマットのファイルから直接行うことができる。また、Cloud Storage からのデータロードも可能である。ビッグデータの解析や絞込みは RDB (Relational Database) の SQL に類似したクエリ言語を使用し、UI 画面や PC のコマンドラインから容易にデータ検索を行うことができる。他にも Excel を使用し検索・表示を行うことが出来る。[3]

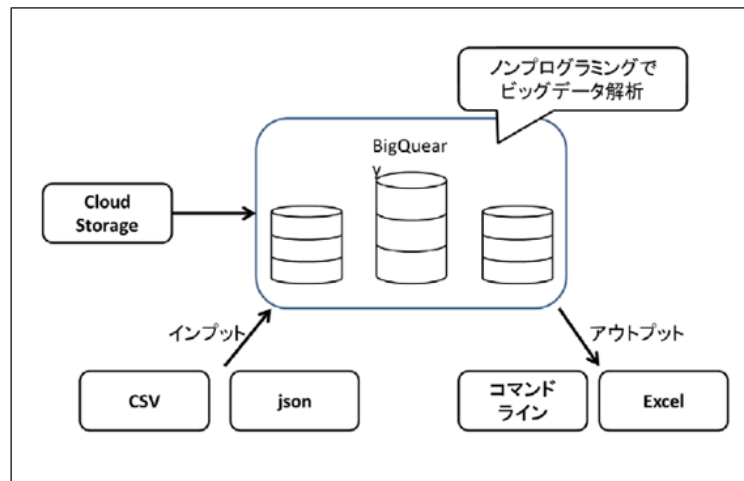


図 2.7 BigQuery の入出力

BigQuery を使用したインタラクティブクエリ処理では、マウス操作と簡単なキー入力によって全ての操作を行うことができる。また結果出力も数秒から数分以内で得ることができる。したがって、何かこのデータを解析したいと考えたとき、その場で気軽に行えるという点が一番の特徴として挙げられる。

BigQuery でのビッグデータ解析は大量のデータを超高速で行えるのも大きな特徴である。例として、15 億行のデータに対する比較的複雑な集計問い合わせが 20 秒から 25 秒で返ってきたというユーザの実行結果もある。BigQuery では、インデックスを作成する必要がなくデータをロードするだけでこのような高速クエリが実行できる。キャッシュは戸鵜ボタンから有効無効を切り替えられるが、キャッシュを使っていなくても、使っているばあ地と同様の結果が得られる。[3]

ストリームデータ処理

ストリームデータ処理は、大量発生する時系列のデータ（ストリームデータ）をリアルタイムに逐次処理する技術。

ストリームデータ処理は、データ発生時に、あらかじめ登録したシナリオにしたがって集計・分析に必要なデータを抽出し、データ処理を行う。その際、分析対象データをメモリー上で処理する「インメモリーデータ処理技術」により、高速なデータ処理を実現している。これらの技術によって、大量データを高速に、かつリアルタイムに処理できる。例えば、株価のテクニカル指標やランキング情報から売買をリアルタイムに自動判定する。といったシステムに大変有効である。他にも、リアルタイムの在庫管理や、不正操作の監視を行うシステムなど、多くの利用目的が考えられる。[3]

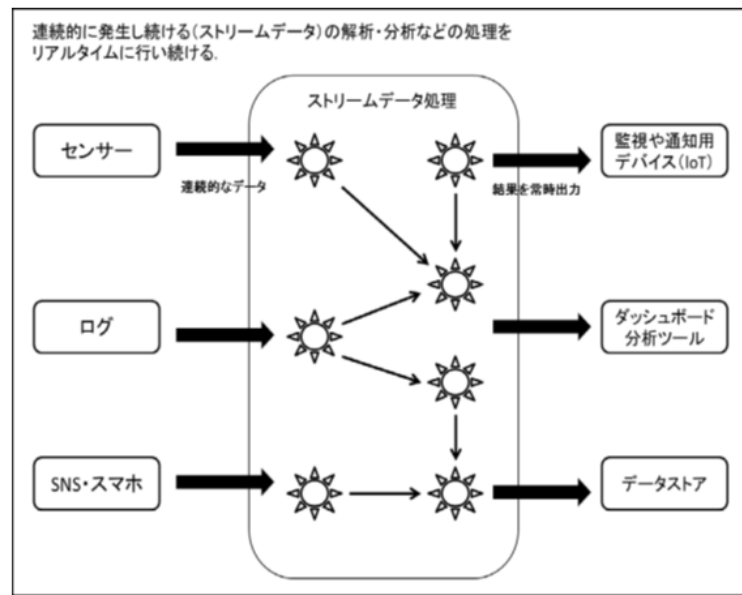


図 2.8 ストリームデータ処理

第 3 章

目的

研究目的

Wikipedia を一つのプロジェクトとみなし、このオンライン百科事典で品質管理がどのように行われているか調査する。この調査により、オープンな共同作業プロジェクトにおける、品質管理マネジメントのあり方についての知見を得たい。

プロジェクトマネジメントとの関連

本研究は、プロジェクトマネジメントを学ぶことを目的としているため、プロジェクトマネジメントとの関連が全面的にある。

考案するゲームは、プロジェクトマネジメント知識体系ガイド (PMBOK®ガイド) の第 4 版(以下、PMBOK)を参考にし、プロジェクトマネジメントとの一連の活動や、PMBOK に記載されている 9 つの知識エリアについての内容を活用する。[4]

PMBOK とは、プロジェクトマネジメントに関する知識体系である。

現在は、PMBOK に従ってプロジェクトマネジメントを実施することが、デファクトスタンダードになっている。

PMBOK に記載されている 9 つの知識エリアとは、何をやるべきかという観点、何を管理するべきかという観点からみたものである。

9 つの知識エリアは、以下 9 つのマネジメントについての内容となっている。

1. プロジェクト統合マネジメント
2. プロジェクト・スコープ・マネジメント
3. プロジェクト・タイム・マネジメント
4. プロジェクト・コスト・マネジメント
5. プロジェクト品質マネジメント
6. プロジェクト人的資源マネジメント
7. プロジェクト・コミュニケーション・マネジメント
8. プロジェクト・リスク・マネジメント
9. プロジェクト調達マネジメント

本研究では、上記の PMBOK の中の品質マネジメントが関連性が最もあるものであるといえる。このオープンなプロジェクトの百科事典は記事の作成や、編集が主に行われて創り上げられており、その成果物がこの百科事典である。

成果物のイメージ 差し戻しに関するデータを収集し、編集回数や頻度などの要素を洗い出す。そして、いくつかの要素から条件を決めクラスター分析を行う。その結果から悪意のある編集がされている記事に共通する点を見つけ、Wikipedia のオープンなプロジェクトでの品質マネジメントの知見を得る。

第 4 章

手法

研究方法

1. Wikipedia 日本語版の編集履歴まで含んだファイルをダウンロードし、ローカルでデータマイニングを行う。
2. どのような品質管理が行われているか、分析結果から調査を行う。
3. オープンなプロジェクトにおける品質管理マネジメントのあり方を提案する。

研究を行うための用意

開発環境として Linux を扱う。そのために、VirtualBox と ubuntu を用意する。

4.1 VirtualBox とは

使用しているパソコン上に仮想的なパソコンを作成し、別の OS をインストール・実行できるフリーのパソコン仮想化ソフトのことである。本研究では、LinuxOS を扱いたいですが、パソコン本体は WindowsOS の為、このソフトを利用する。

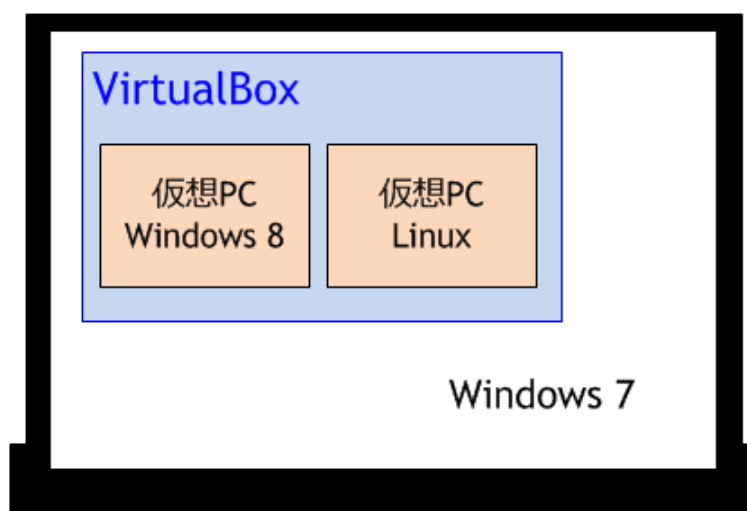


図 4.1 図の挿入例

VirtualBox はコンピュータ上で直接動作している通常の OS にとってはアプリケーションの一つであり、他のソフトと同じように起動することができる。起動すると仮想的なコンピューターが構築され、元の OS とは独立に別の OS を起動することができる。VirtualBox が実行されている OS をホスト OS、VirtualBox 上で実行されている OS をゲスト OS という。

元は独立系のソフトウェア企業が開発・販売していた製品だった。しかし、開発元が Sun Microsystems 社に買収され、その後同社が Oracle 社に買収されたため、Oracle 社が開発元となり、正式名称も「Oracle VM VirtualBox」となった。また、VirtualBox 本体は GPL に基づいたオープンソースソフトウェアとして公開され、誰でも自由に入手・利用・改変・再配布などが行える。

VirtualBox を使う上での注意点

現時点での VirtualBox は仮想メモリをサポートしていないため、実メモリ以上のメモリを仮想 PC が使用することはできない。仮想メモリを使うと動作が遅くなるため、仮想 PC には実メモリ以内のサイズを割り当てる。そのため、仮想 PC を 1 台だけ起動するのであれば問題ないが、複数の仮想 PC を同時に起動させる場合これがネックになってしまう。同時起動させる全ての仮想 PC のメモリサイズの合計が実メモリのサイズを超えないようにする。そのため、VirtualBox をインストールする PC には多くのメモリが必要で、最低 4GB 以上の PC を使うようにするべきである。

4.2 VirtualBox のインストール

4.2.1 ダウンロード

Oracle が提供している OracleVirtualBox というのを下記のサイトからダウンロードする。本研究では WindowsOS を使用しているので、VirtualBox platform packages の中にある「VirtualBox 5.0.4 for Windows hosts」というのを選択する。



図 4.2 ダウンロードサイト

ダウンロードサイト

<https://www.virtualbox.org/wiki/Downloads>

4.2.2 インストーラーの実行

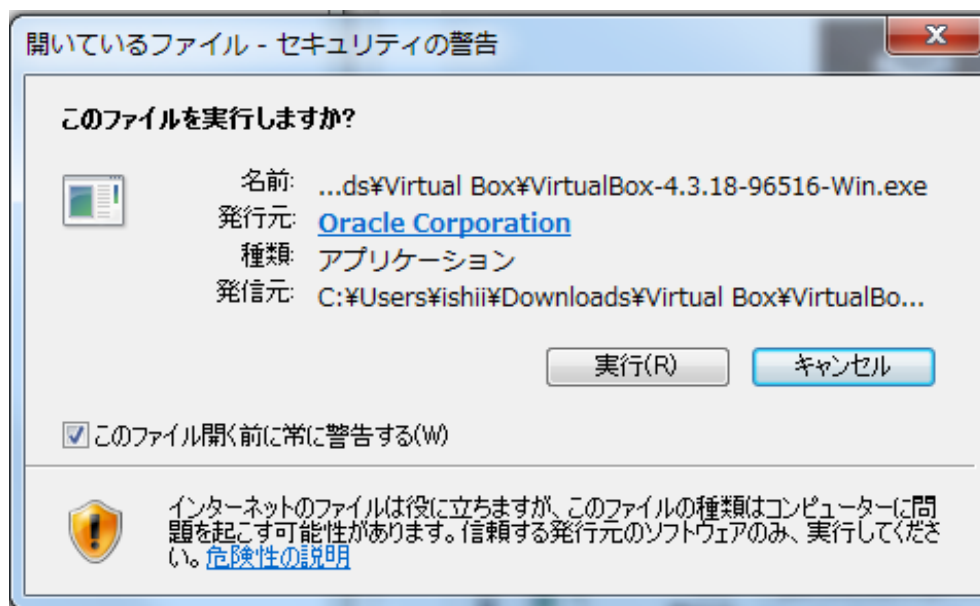


図 4.3 警告画面

インストーラーを起動し、セットアップウィザードを起動する。「セキュリティの警告」ダイアログが表示された場合は、「実行」をクリックする。



図 4.4 セットアップ画面 1

右下の「Next」をクリックする。

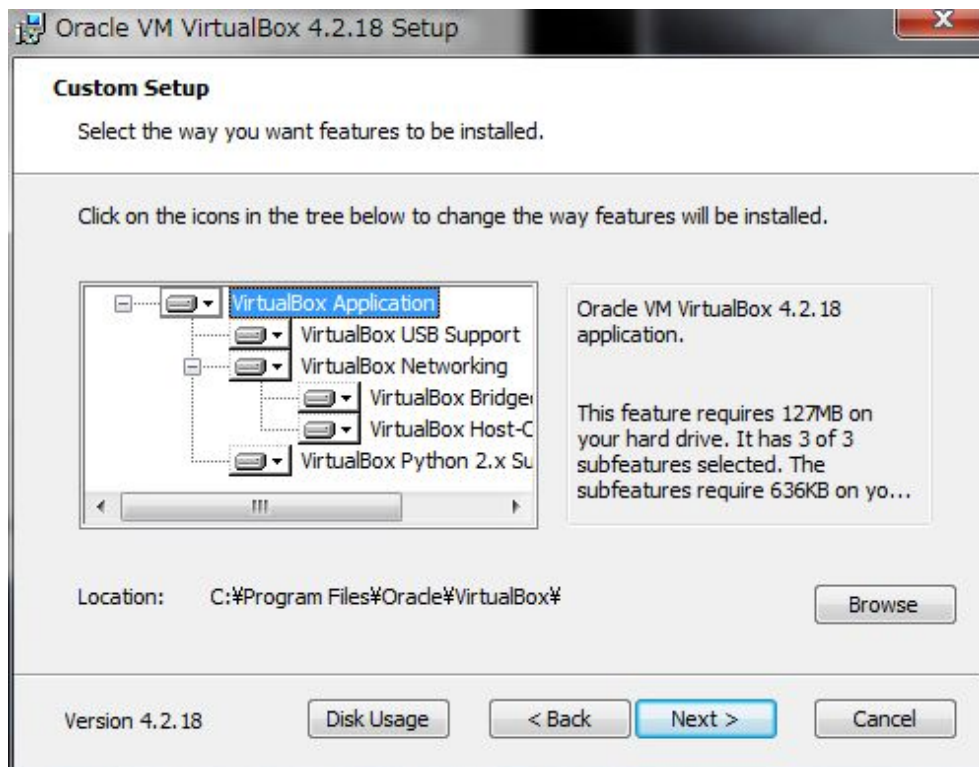


図 4.5 セットアップ画面 2

VirtualBox Application を選択したまま、「Next」をクリックする。

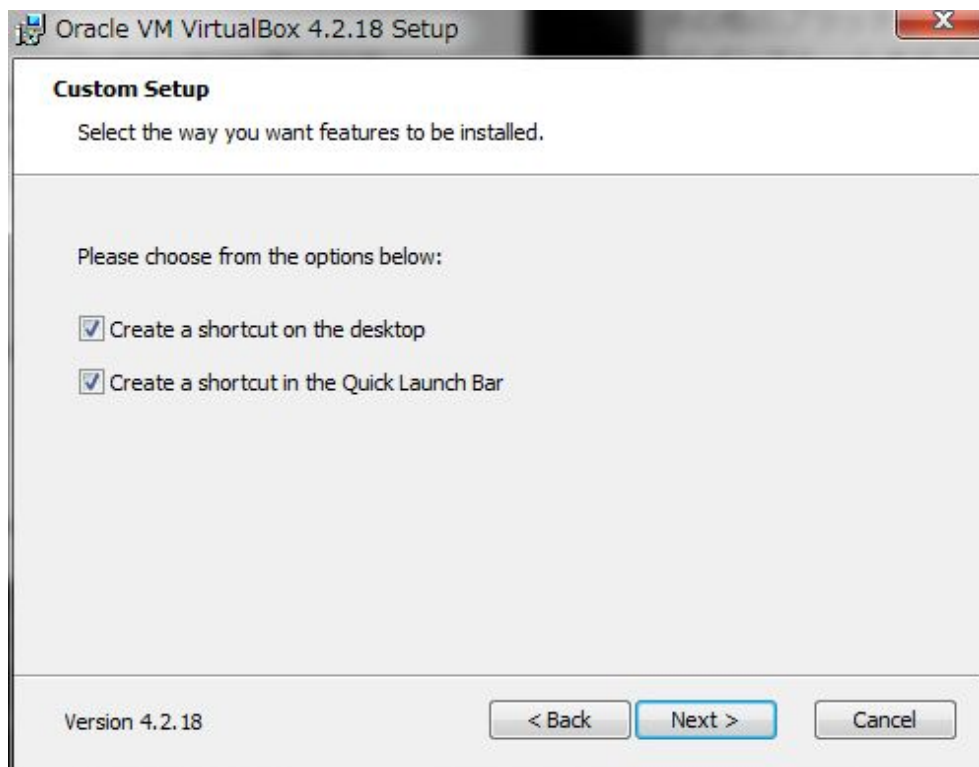


図 4.6 セットアップ画面 3

2 つともチェックマークをつけたまま、「Next」をクリックする。



図 4.7 セットアップ画面 4

Warning:Network Interfaces 画面が表示されたら、「Yes」をクリックする。

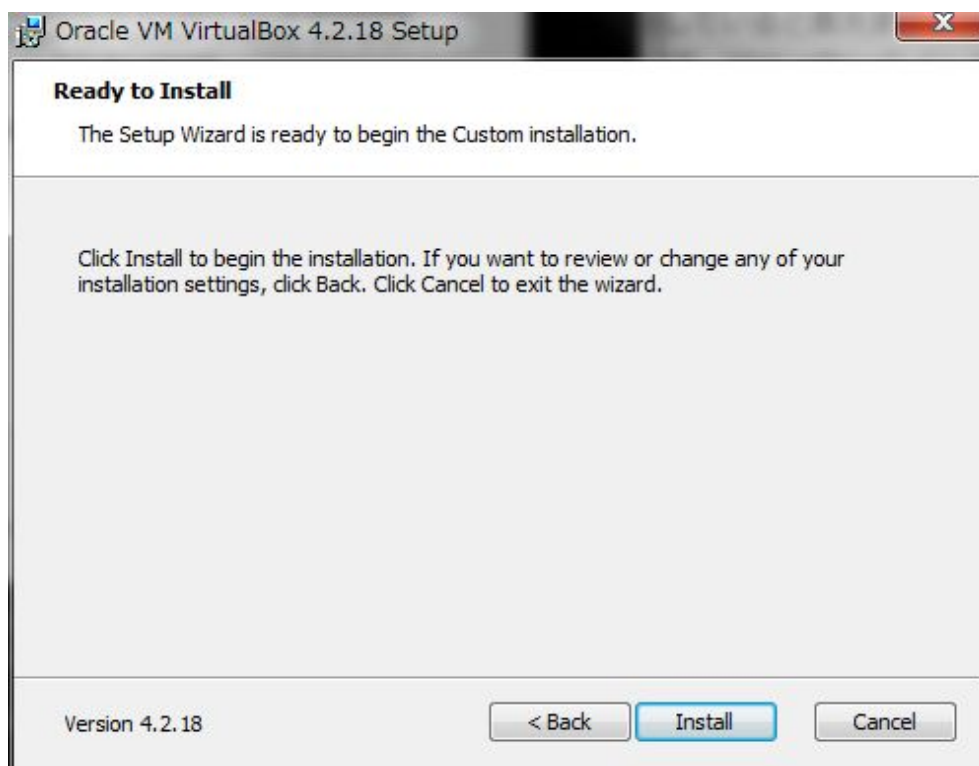


図 4.8 セットアップ画面 5

Ready to Install 画面が表示されたら、「Install」をクリックする。

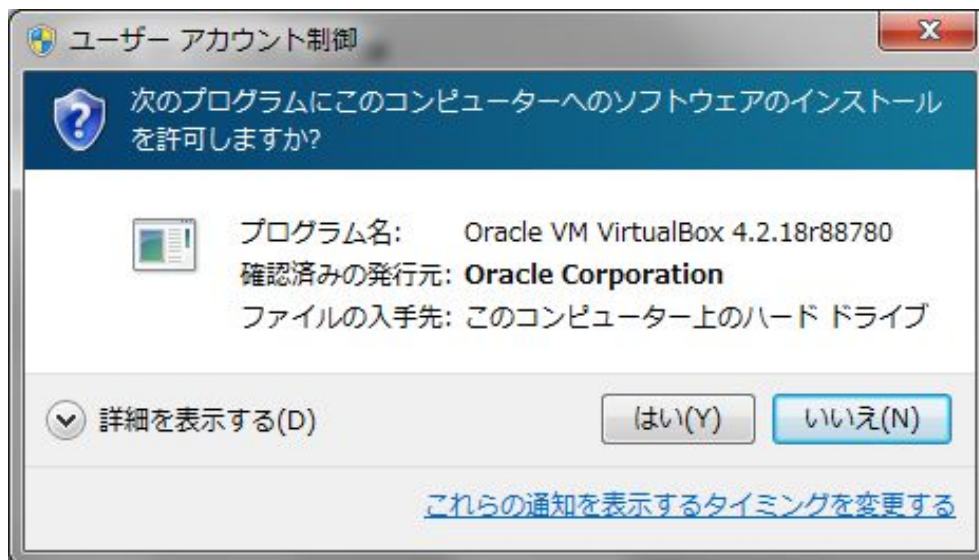


図 4.9 ユーザーアカウント制御画面

インストール中にユーザーアカウント制御によって、ソフトウェアのインストールを許可する必要がある。その場合は画面が表示されるので、「はい」をクリックする。

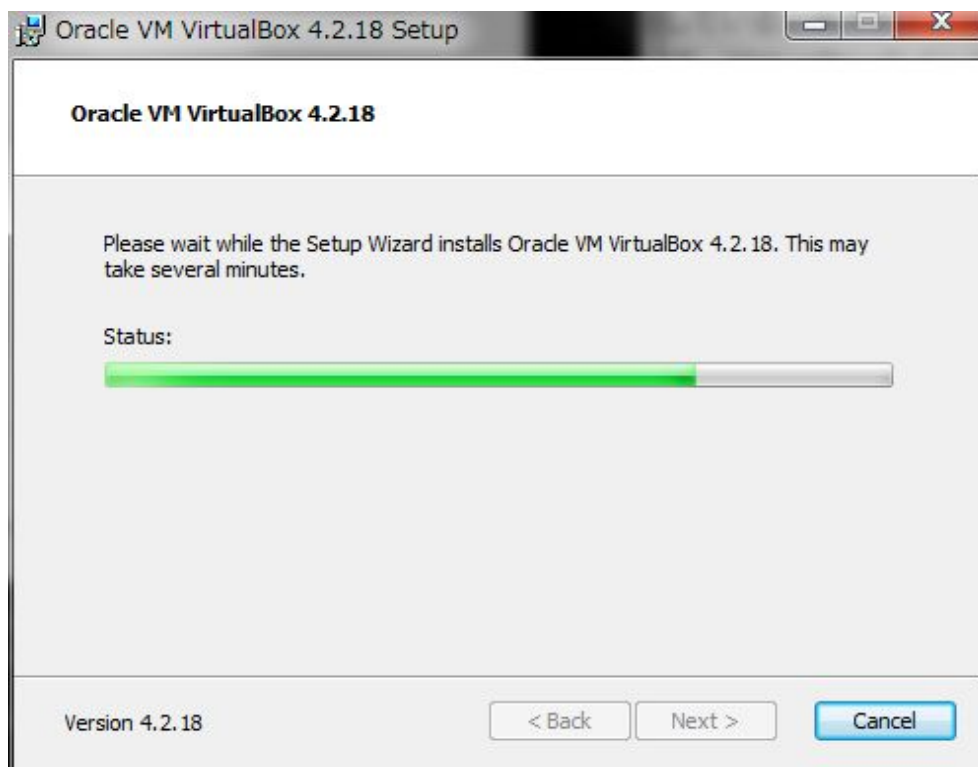


図 4.10 インストール中の画面

インストール実行画面になるので、終了するまでそのままにする。

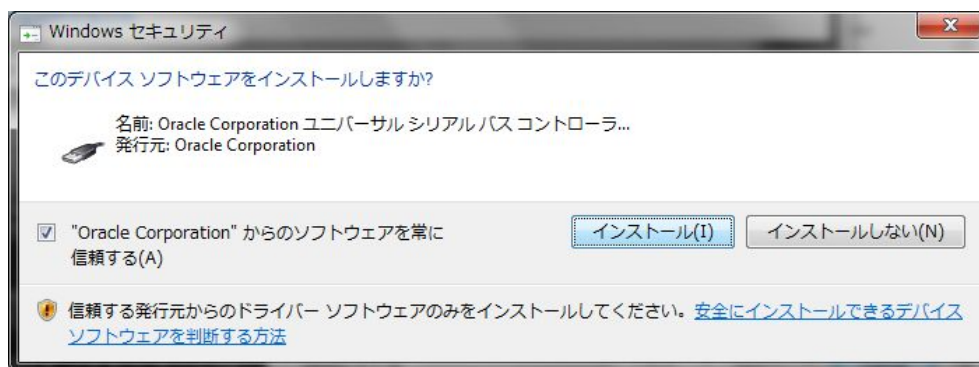


図 4.11 Windows セキュリティの画面

Windows セキュリティの画面が表示されたら、「'OracleCorporation' からのソフトウェアを常に信頼する」というところにチェックし、インストールをクリックする。



図 4.12 インストール完了の画面

インストールが完了すると、インストール後に VirtualBox を起動するか聞かれる。起動してもしなくても構わない。

「Finish」をクリックすれば、インストール完了。



図 4.13 VirtualBox 起動時の画面

インストールした「VirtualBox.exe」を起動して正常に動くか確認する。

4.3 用語

本節では、本研究で利用する VirtualBox の用語について説明する。

4.3.1 ホストマシン（物理マシン）

物理的に存在するコンピュータのこと。

4.3.2 ホスト OS

ホストマシンにインストールされている OS のこと。VirtualBox はホスト OS にインストールされる。

4.3.3 パーチャルマシン（仮想マシン）

VirtualBox が作成する論理的なマシンのこと。ゲストマシンに割り当てるために、VirtualBox がホストマシンのコンピュータ資源（CPU やメモリ、HDD 等）の一部を仮想化する。ホストマシンの資源を使いきらない限り、ゲストマシンを複数作成したり、多重起動させることができる。

4.3.4 ゲスト OS（仮想 OS）

ゲストマシンにインストールされる OS のこと。本研究では、Ubuntu というものをインストールする。

4.3.5 仮想ディスク

ゲストマシンが使用する仮想のハードディスクのこと。バーチャルマシンからはこれを物理ディスクとして扱うことができる。仮想ディスクの実態はホストマシン内にファイルとして存在する。

4.4 Ubuntu とは

Ubuntu（ウブントゥ）とは、コミュニティにより開発されているオペレーティングシステムのこと。ラップトップ、デスクトップ、そしてサーバーに利用することができる。Ubuntu には、家庭・学校・職場で必要とされるワープロやメールソフトから、サーバーソフトウェアやプログラミングツールまで、あらゆるソフトウェアが含まれています。

Ubuntu は現在、そして将来に渡って無償で提供されます。ライセンス料を支払う必要はありません。Ubuntu をダウンロードすれば、友達や家族と、あるいは学校やビジネスに、完全に無料で利用できます。

提供会社は、新しいデスクトップおよびサーバーを 6 ヶ月ごとにリリースすることを宣言している。これにより、オープンソースの世界が提供する最新の優れたアプリケーションを常に利用できる。

4.5 Ubuntu のインストール

4.5.1 ISO イメージをダウンロードする

まず、<https://www.ubuntulinux.jp/download/ja-remix> より Ubuntu 14.04 の ISO イメージをダウンロードする。本研究では 64bit 版を選択した。



図 4.14 Ubuntu の ISO イメージダウンロードページ

4.5.2 インストールを開始する

次にインストールした VirtualBox を立ち上げ、ウインドウ左上にある「新規 (N)」のボタンを押してゲストマシンの作成を行う。ゲストマシンの名前、メモリサイズ、HDD の設定をするとウインドウが閉じられる。

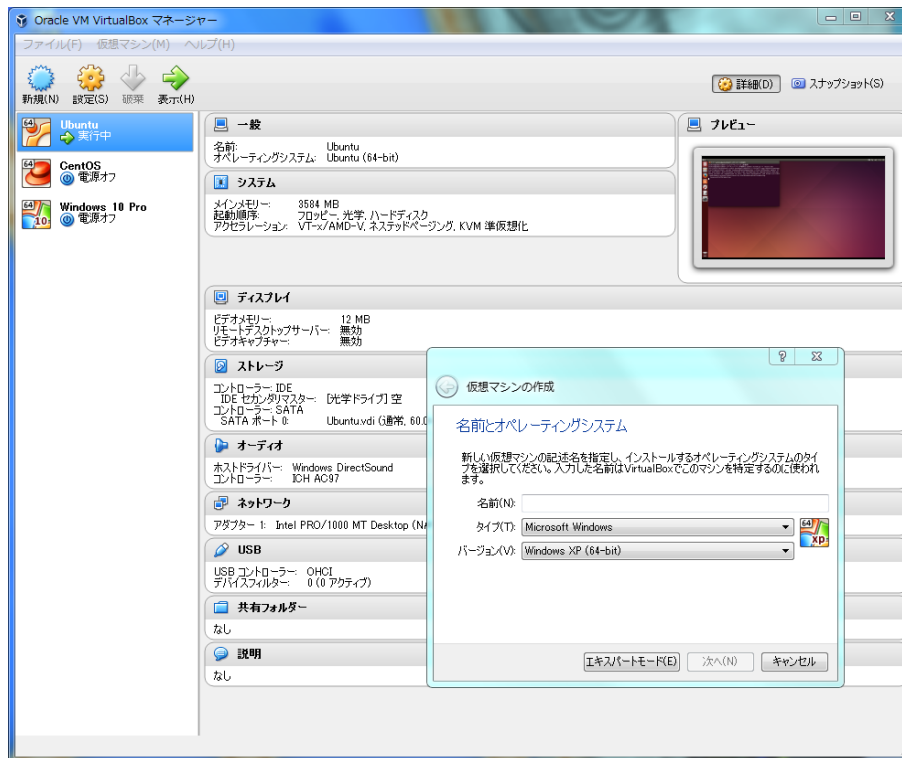


図 4.15 VirtualBox のウィンドウとゲストマシン作成ウィンドウ

続いて「設定 (S)」を開き、「ストレージ」の項目からダウンロードした ISO ファイルをセットして「OK」をクリックする。

「起動 (T)」を押すとゲストマシンが起動し、Ubuntu のインストールウィザードが表示される。

表示内容に従ってウィザードを進めていくと、ゲストマシン内に Ubuntu がインストールされる。

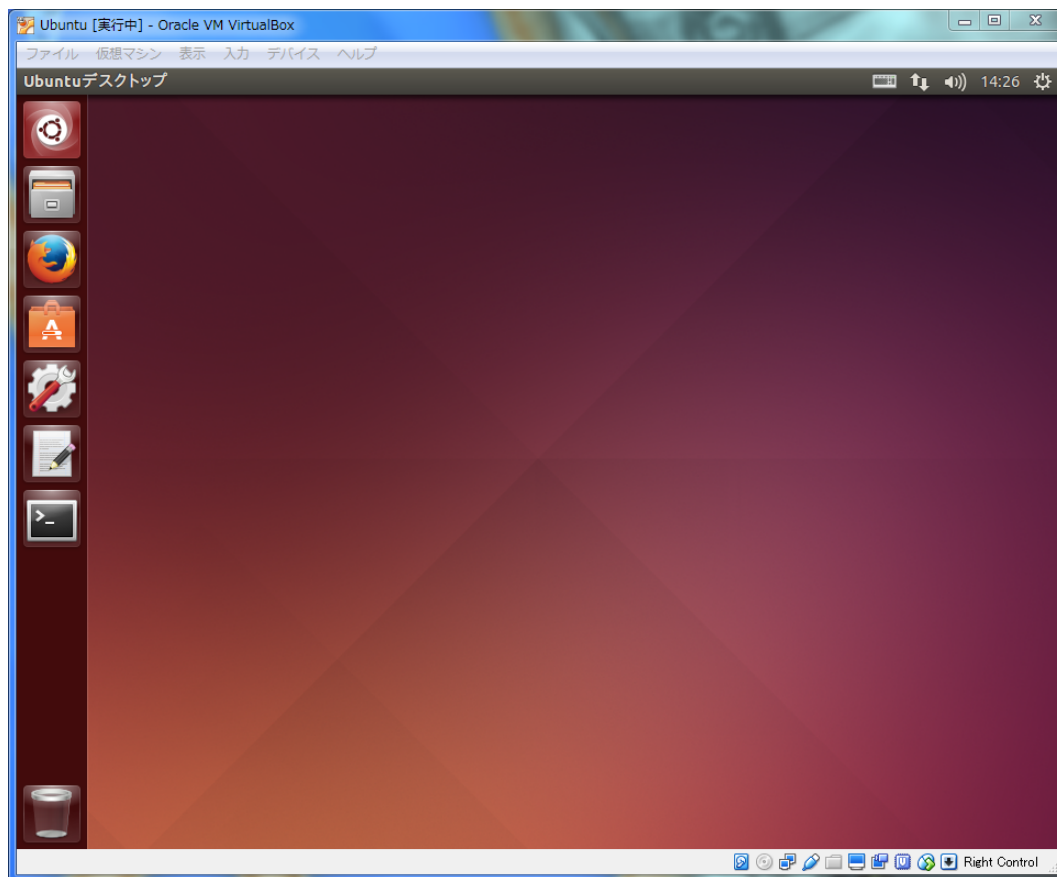


図 4.16 VirtualBox 上で動作する Ubuntu

4.6 端末

本研究では、ダウンロードしたファイルを Ubuntu の端末内で扱う。
起動方法はいろいろあるが、2 つ記載する

1. ショートカットキーで起動する。

ubuntu の画面を開いた状態で「Ctrl+Alt+T」を押す。このショートカットキーを押すだけで、端末が開く。

2. コンピュータとオンラインソースを検索」から起動する

Launcher にある「Ubunutu ソフトウェアセンター」を開く。「インストール済み」を選択する。検索ワード欄に「端末」または「ta」と入力する。カテゴリの中に「端末 (gnome-terminal)」があるので、それを選択する。

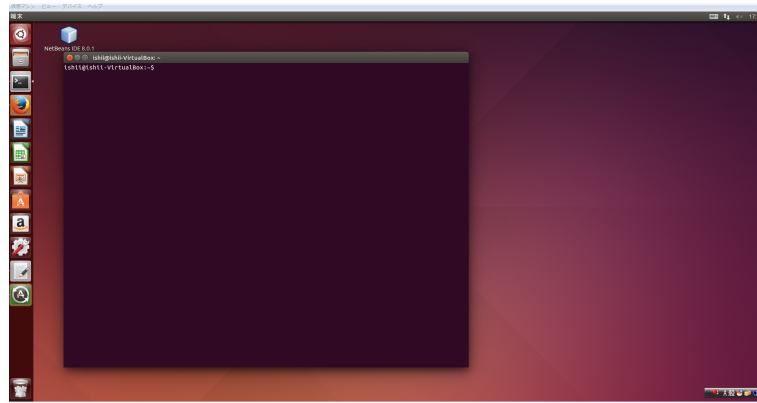


図 4.17 端末を起動した画面

4.7 MySQL

MySQL の操作 ,そこで使用した SQL 文の説明を行う . MySQL の操作は ,Ubuntu の端末内と phpmyadmin で行う .

4.7.1 MySQL へのインストール

Ubuntu では , 以下のコマンドで MySQL をインストールを行う . 途中でパスワードを聞かれたら , 「pass」のような簡単なものを設定する .

```
mysql>sudo apt-get install mysql-server mysql-client
```

4.7.2 MySQL への接続

端末から接続を行う . 端末内で次のように入力すれば MySQL に接続できる .

```
mysql -u ユーザ名 -p パスワード --default-character-set=文字コード
```

本研究では , ユーザ名は 「root」 , パスワードは 「yasu0705」 を入力する . またデータのインポートする際に LOAD 文を使いたいので , MySQL の接続時に 「mysql --local-infile . . . 」 と入力する .

以下のように接続を行う .

```
mysql>mysql -uroot -pyasu0705 --local-infile --default-character-set=utf8
```

```
ishii@ishii-VirtualBox: ~  
ishii@ishii-VirtualBox:~$ mysql -uroot -pyasu0705 --local-infile --default-character-set=utf8  
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 877  
Server version: 5.5.46-0ubuntu0.14.04.2 (Ubuntu)  
  
Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
mysql>
```

図 4.18 MySQL 接続画面

4.7.3 phpmyadmin

PHP で実装された MySQL の管理ツール。MySQL のデータベースやテーブルの作成を行ったり、データの追加や参照などを作成することなくブラウザから行うことができる。

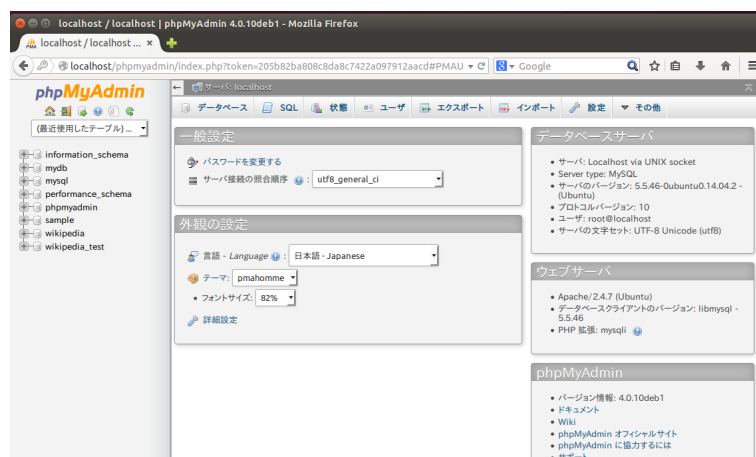


図 4.19 phpmyadmin のメインページ

4.8 Wikipedia の編集履歴データの取得

4.8.1 ファイルのダウンロード

日本語版 Wikipedia のデータは、<https://dumps.wikimedia.org/jawiki/>からダウンロードできる。

Index of /jawiki/

../		
20150221/	24-Feb-2015 17:51	-
20150313/	18-Mar-2015 14:37	-
20150402/	05-Apr-2015 08:19	-
20150422/	25-Apr-2015 13:52	-
20150512/	15-May-2015 08:17	-
20150602/	18-Jun-2015 01:34	-
20150703/	08-Jul-2015 14:44	-
20150805/	13-Aug-2015 21:29	-
20150826/	29-Aug-2015 04:03	-
20150901/	10-Sep-2015 23:02	-
20151002/	06-Oct-2015 03:14	-
latest/	06-Oct-2015 03:14	-

図 4.20 日本語版 Wikipedia データダウンロードサイト

再現性を確保するため、20150901 番を扱う。latest だと、最新だが、日々更新されて変更されてしまう可能性があるため今回は利用しない。

データの中身にどのようなものが調べた。[5] 調べた結果、stub-meta-history.xml が使えそうなので、これを利用する。また、「stub-meta-history1」などの分割ファイルも利用する。

2015-09-03 20:46:56	done	All pages, current versions only. jawiki-20150901-pages-meta-current1.xml.bz2 318.6 MB jawiki-20150901-pages-meta-current2.xml.bz2 687.5 MB jawiki-20150901-pages-meta-current3.xml.bz2 263.3 MB jawiki-20150901-pages-meta-current4.xml.bz2 1.2 GB
2015-09-03 04:50:23	done	Recombine articles, templates, media/file descriptions, and primary meta-pages. jawiki-20150901-pages-articles.xml.bz2 2.0 GB
2015-09-03 01:08:10	done	Articles, templates, media/file descriptions, and primary meta-pages. jawiki-20150901-pages-articles1.xml.bz2 292.6 MB jawiki-20150901-pages-articles2.xml.bz2 587.7 MB jawiki-20150901-pages-articles3.xml.bz2 216.1 MB jawiki-20150901-pages-articles4.xml.bz2 974.8 MB
2015-09-02 10:10:48	done	Recombine first-pass for page XML data dumps <i>These files contain no page text, only revision meta data.</i> jawiki-20150901-stub-meta-history.xml.gz 3.9 GB jawiki-20150901-stub-meta-current.xml.gz 271.1 MB jawiki-20150901-stub-articles.xml.gz 197.2 MB
2015-09-02 04:54:26	done	First-pass for page XML data dumps <i>These files contain no page text, only revision metadata.</i> jawiki-20150901-stub-meta-history1.xml.gz 894.5 MB jawiki-20150901-stub-meta-history2.xml.gz 1.4 GB jawiki-20150901-stub-meta-history3.xml.gz 416.8 MB jawiki-20150901-stub-meta-history4.xml.gz 1.2 GB jawiki-20150901-stub-meta-current1.xml.gz 12.4 MB jawiki-20150901-stub-meta-current2.xml.gz 64.9 MB jawiki-20150901-stub-meta-current3.xml.gz 33.6 MB jawiki-20150901-stub-meta-current4.xml.gz 160.2 MB jawiki-20150901-stub-articles1.xml.gz 10.6 MB jawiki-20150901-stub-articles2.xml.gz 50.5 MB jawiki-20150901-stub-articles3.xml.gz 25.1 MB jawiki-20150901-stub-articles4.xml.gz 110.9 MB
2015-09-08 18:41:15	done	Recombine extracted page abstracts for Yahoo jawiki-20150901-abstract.xml 1.7 GB
2015-09-08 18:40:39	done	Extracted page abstracts for Yahoo 2015-09-08 18:40:34: jawiki (10 7400) 8393 pages (31.7 2258.3/sec all curr), 8393 revs (31.7 105.7/sec all curr), ETA 2015-09-09 04:34:51 [max 3277733] jawiki-20150901-abstract1.xml 614.1 MB jawiki-20150901-abstract2.xml 396.7 MB jawiki-20150901-abstract3.xml 379.4 MB jawiki-20150901-abstract4.xml 349.0 MB
2015-09-08 16:33:56	done	List of all page titles jawiki-20150901-all-titles.gz 16.8 MB
2015-09-08 16:33:46	done	List of page titles in main namespace jawiki-20150901-all-titles-in-ns0.gz 9.4 MB
2015-09-02 13:43:57	done	List of pages' geographical coordinates jawiki-20150901-geo-tags.sql.gz 1.3 MB

図 4.21 20150901 の stub-meta-history

4.9 XML ファイル

Wikipedia のデータは XML というものだった。XML とは、ExtensibleMarkupLanguage の略であり、インターネット上でさまざまなデータを扱う倍に利点があるファイルである。1998 年にでた比較的新しい言語だが、仕様が簡単のため、広く使用されるようになった。

XML 文書はテキストファイルの形で存在するため、そのままソフトで利用することが出来ない。個々のソフトの中に XML 文書を解釈するプログラムを持たせる方法があるが、XML は一定の形式が定められているため、汎用的な解釈プログラムである XML パーサーによって変換したデータをソフトが使うようにした方が効率が良い。アプリケーションソフトで XML 文書を扱う場合には、直接 XML 文書を読むのではなく、XML パーサを扱うのが一般的になっている。[6]

4.10 XML パーサ

XML 文書を、アプリケーションソフトが利用しやすい形に変換するソフトウェアのこと。変換時に、XML 文書が文法に照らして性格に記述されているかどうかを同時に検証するものである。

XML パーサの中には DOM(Document Object Model) と、SAX(Simple API for XML) という最も広く利用されている標準的な API がある。

本研究では、この XML の文書の操作には、SAX パーサというものをを用いて扱う。Wikipedia の編集履歴データは膨大な為、木構造として全てのデータを読み込んでから処理する DOM パーサでは、データの扱いが困難なためである。

4.10.1 DOM パーサ

DOM とは、XML 文書や HTML 文書を構成する要素をコンピュータプログラムで参照したり操作したりするための取り決め (API) の 1 つである。

HTML や XML で記述された Web ページなどの構成要素 (見出し、段落、領域、画像、リンクなど) と、それらの配置は見栄えなどを定めた属性情報などを参照、制御する手法を定めている。Web ブラウザなどに実装されており、ページ上に JavaScript など記述されたスクリプトからページ内の各要素を読み取ったり、内容や設定の変更、要素の追加や削除などを行う標準的な手段として用いられる。

文書を DOM で表したデータは、文書の最上位の要素を頂点として、各要素が枝分かれしていく木構造 (ツリー構造) となっており、これを「DOM ツリー」と呼ぶこともある。[7]

4.10.2 SAX パーサ

XML 文書の解釈や検証を行う「XML パーサー」というプログラムを利用する際に使う利用手順の 1 つである。また、DOM と並んで最も広く利用されている API の 1 つとして使われる。

XML 文書を一つの木構造に変換する DOM と違って、XML 文書を先頭から一行ずつ読み込んで、要素が現れるたびに対応する処理手順を呼び出すという方式を用いている。よって巨大な XM を扱ってもメモリに負担がかからず高速に処理できるという特徴がある。その反面、文書の構造を自由にたどれないので、処理の柔軟性に劣り、複雑な処理には向かない。[8]

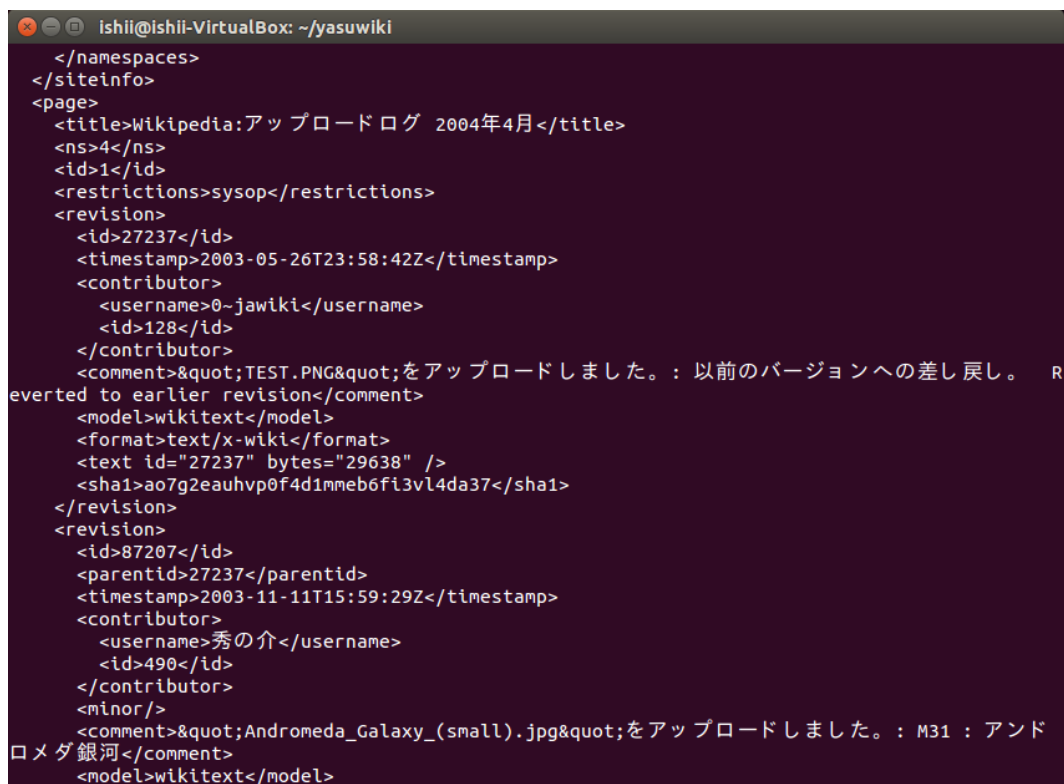
第 5 章

結果

5.1 Wikipedia の履歴の調査

ダウンロードしたファイルの中身がどんななのか確認するため、以下のコマンドを入力する。q ボタンを押すと終了する。

```
gunzip -c jawiki-20150901-stub-meta-history.xml.gz | less
```



```

</namespaces>
</siteinfo>
<page>
<title>Wikipedia:アップロードログ 2004年4月</title>
<ns>4</ns>
<id>1</id>
<restrictions>sysop</restrictions>
<revision>
<id>27237</id>
<timestamp>2003-05-26T23:58:42Z</timestamp>
<contributor>
<username>0~jawiki</username>
<id>128</id>
</contributor>
<comment>&quot;TEST.PNG&quot;をアップロードしました。: 以前のバージョンへの差し戻し。 R
everted to earlier revision</comment>
<model>wikitext</model>
<format>text/x-wiki</format>
<text id="27237" bytes="29638" />
<sha1>ao7g2eauhvp0f4d1mmeb6fi3vl4da37</sha1>
</revision>
<revision>
<id>87207</id>
<parentid>27237</parentid>
<timestamp>2003-11-11T15:59:29Z</timestamp>
<contributor>
<username>秀の介</username>
<id>490</id>
</contributor>
<minor/>
<comment>&quot;Andromeda_Galaxy_(small).jpg&quot;をアップロードしました。: M31 : アンド
ロメダ銀河</comment>
<model>wikitext</model>

```

図 5.1 less で中身確認

5.1.1 展開

展開する際は、「-k」をつける。wikipedia のダウンロードファイルは時間がかかるため、一度ダウンロードしたら消さずに取っておく。

```
gunzip -k jawiki-20150901-stub-meta-history.xml.gz
gunzip -k jawiki-20150901-stub-meta-history{1,2,3,4}.xml.gz
```

5.1.2 revision 数

ページごとの revision 数を求める。詳細は以下の revisions.py を参照。

revisions.py

```
# -*- coding: utf-8 -*-

import xml.sax
import sys

class myHandler(xml.sax.ContentHandler):
    reading = {"title": False}
    title = ""
    revisions = 0

    def __init__(self):
        xml.sax.ContentHandler.__init__(self)

    def startElement(self, name, attrs):
        self.reading[name] = True

    def endElement(self, name):
        self.reading[name] = False
        if name == "page":
            print(self.title + "\t" + str(self.revisions))
            self.title = ""
            self.revisions = 0
        elif name == "revision":
            self.revisions = self.revisions + 1

    def characters(self, content):
        if self.reading["title"] == True:
            self.title = self.title + content

def main():
    xml.sax.parse(sys.stdin, myHandler())

if __name__ == "__main__":
    main()
```


以下のコマンドを入力する .

```
cat jawiki-20150901-stub-meta-history.xml | python3 revisions.py | less
```

```

ishii@ishii-VirtualBox: ~/yasuwiki
Wikipedia:アップロードログ 2004年4月 6
Wikipedia:削除記録/過去ログ 2002年12月 1
アンパサンド 114
Wikipedia:Sandbox 17
利用者:Brion VIBBER 29
言語 598
日本語 2270
地理学 334
EU (曖昧さ回避) 58
ノート:メインページ/過去ログリスト / -2004年9月上旬 618
利用者:Youssefsan 2
利用者:Aoineko 15
利用者:Alan D 1
国の一覧 713
Wikipedia:LanguageJa.php 50
SandBox 9
パリ 854
ヨーロッパ 613
利用者:Xaos 7
利用者:Hijiri 30
生物 314
コケ植物 120
利用者-会話:Hijiri 13
利用者-会話:Aoineko 18
HomePage 2
社会学 307
古代エジプト 370
エジプト 809
Wikipedia:About 110
Wikipedia:Text of GNU Free Documentation License 18
著作権の保護期間 269
台東区 772
地理 84
生物学 450
社会 202
:
```

図 5.2 revisions 数の確認

データを取る際は「| less」の部分を「> ファイル名」にする . 本研究では , 以下のように取得した .

```
cat jawiki-20150901-stub-meta-history.xml | python3 revisions.py > revisions.dat
```

次に取得したデータから , revision 数が多いページを見つける . 詳細は以下の revisions.sh を参照 .

```
revisions.sh
```

```
cat revisions.dat | sort -r -n -k 2 -t " " > sort.dat ( ""の中は TAB )
```

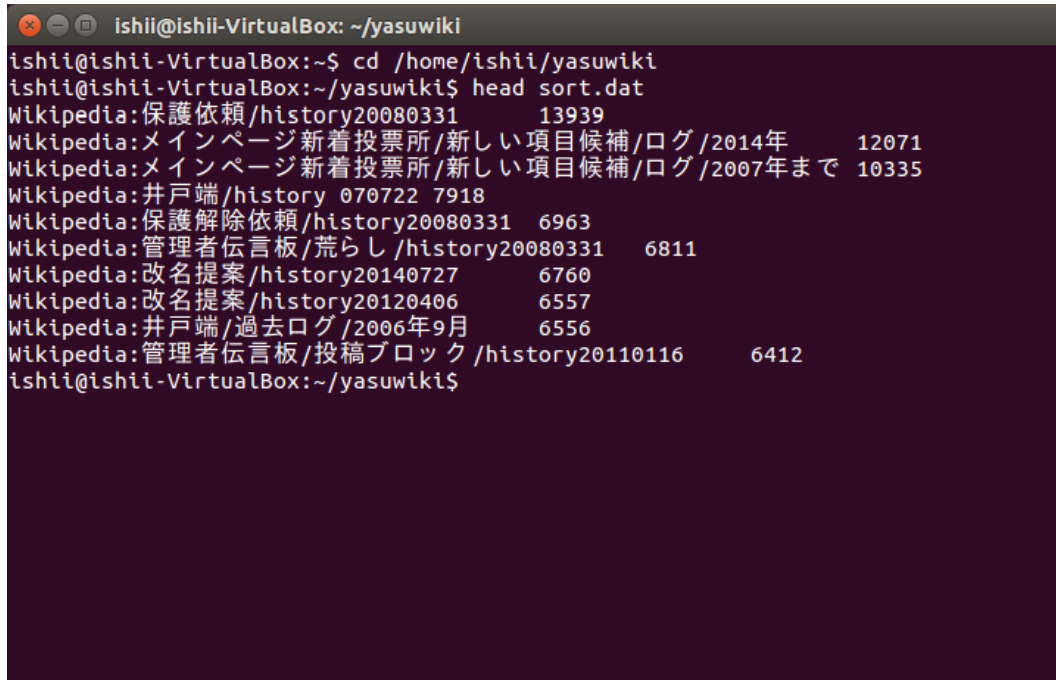
以下のコマンドを入力する .

```
time sh revisions.sh
```

抽出したデータを確認する為、以下のコマンドを入力する。

```
head sort.dat
```

結果は以下のとおり



```

ishii@ishii-VirtualBox: ~/yasuwiki
ishii@ishii-VirtualBox:~$ cd /home/ishii/yasuwiki
ishii@ishii-VirtualBox:~/yasuwiki$ head sort.dat
Wikipedia:保護依頼/history20080331      13939
Wikipedia:メインページ新着投票所/新しい項目候補/ログ/2014年      12071
Wikipedia:メインページ新着投票所/新しい項目候補/ログ/2007年まで  10335
Wikipedia:井戸端/history 070722 7918
Wikipedia:保護解除依頼/history20080331  6963
Wikipedia:管理者伝言板/荒らし/history20080331  6811
Wikipedia:改名提案/history20140727      6760
Wikipedia:改名提案/history20120406      6557
Wikipedia:井戸端/過去ログ/2006年9月     6556
Wikipedia:管理者伝言板/投稿ブロック/history20110116  6412
ishii@ishii-VirtualBox:~/yasuwiki$

```

図 5.3 revisions 数の多いページ

5.1.3 並列化

wikipedia の編集履歴データには、「jawiki-20150901-stub-meta-history1.xml」などの分割ファイルがある。これらを扱うことを行う。

準備として、以下のコマンドを入力し、page 要素の途中では分割されていないことを確認する。「stub-meta-history」の分割ファイルは 4 つあるので、4 つとも確認する。

```
tail jawiki-20150901-stub-meta-history1.xml
```

ファイルの並列化をして、revision 数を求める。以下のコマンドを入力する。

```

cat jawiki-20150901-stub-meta-history1.xml | python3 revisions.py > revisions1.dat &
cat jawiki-20150901-stub-meta-history2.xml | python3 revisions.py > revisions2.dat &
cat jawiki-20150901-stub-meta-history3.xml | python3 revisions.py > revisions3.dat &
cat jawiki-20150901-stub-meta-history4.xml | python3 revisions.py > revisions4.dat &

```

次に結合してソートするため、以下のコマンドを入力する。

```
cat revisions1,2,3,4.dat | sort -r -n -k 2 -t " " > sort-parallel.dat
```

データを抽出したら「head sort-parllel.dat」で結果を確認する。

「diff sort.dat sort-parallel.dat」で、一括データと分割データの違いを見る。結果が空、つまり両者に違いが無ければ同じデータが取れている。

5.1.4 差し戻し回数

「stub-meta-history」のデータの中には、SHA1 が入っていた。SHA1 とは、認証やデジタル署名などに使われるハッシュ関数のひとつである。2 の 62 条ビット以下の原文から 160 ビットの「ハッシュ値」を生成し、通信経路の両端で比較することで、通信途中で原文が改ざんされていないかを検出することが出来る。

この SHA1 が同じなら差し戻しとみなすことにする。ただし、以下の問題はある。

- テキストが違ってても SHA1 が同じになることはある。
- 差し戻してそのまま編集された場合をかうんとできない。

SHA1 を単純にか添えた結果が編集回数にする。そこから重複を削除した結果が差し戻し以外の回数とし、両方の差が、ここでいう差し戻しの解すという定義にする。詳細は以下の revert.py を参照。

revert.py

```
# -*- coding: utf-8 -*-

import xml.sax
import sys

class myHandler(xml.sax.ContentHandler):
    reading = {"title":False, "sha1":False}
    title = ""
    revisions = 0
    sha1 = ""
    sha1s = set() #SHA1 を数える (重複なし)

    def __init__(self):
        xml.sax.ContentHandler.__init__(self)

    def startElement(self, name, attrs):
        self.reading[name] = True

    def endElement(self, name):
        self.reading[name] = False
        if name == "page":
            revert = self.revisions - len(self.sha1s) #差し戻し回数
            print(self.title + "\t" + str(self.revisions) + "\t" + str(len(self.sha1s)) + "\t" + str(revert))
            self.title = ""
            self.revisions = 0
            self.sha1s.clear() #SHA1 の集合をクリア
        elif name == "sha1":
            self.revisions = self.revisions + 1
            self.sha1s.add(self.sha1)
            self.sha1 = ""
```

```

def characters(self, content):
    if self.reading["title"] == True:
        self.title = self.title + content
    elif self.reading["sha1"] == True:
        self.sha1 = self.sha1 + content

def main():
    xml.sax.parse(sys.stdin, myHandler())

if __name__ == "__main__":
    main()

```

以下のコマンドを入力する。

```

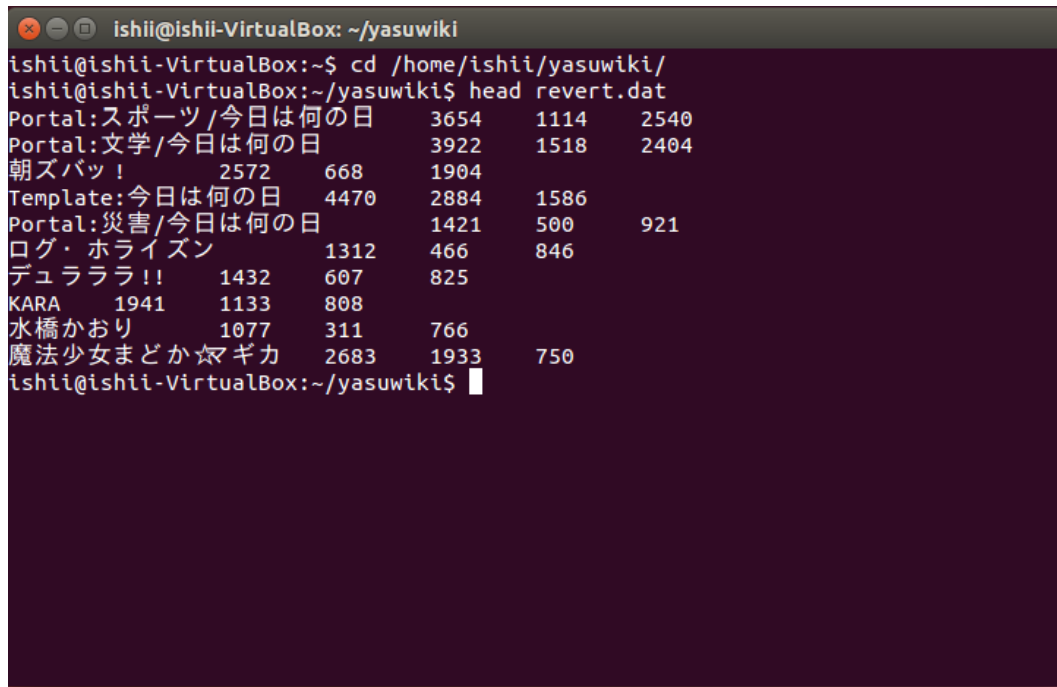
cat jawiki-20150901-stub-meta-history1.xml | python3 revert.py > revert1.dat &
cat jawiki-20150901-stub-meta-history2.xml | python3 revert.py > revert2.dat &
cat jawiki-20150901-stub-meta-history3.xml | python3 revert.py > revert3.dat &
cat jawiki-20150901-stub-meta-history4.xml | python3 revert.py > revert4.dat &

```

次に抽出したデータを一括し、ソートする。

```
cat revert1,2,3,4.dat | sort -r -n -k 4 -t " " > revert.dat
```

「head revert.dat」の結果は以下のとおり。



```

ishii@ishii-VirtualBox: ~/yasuwiki
ishii@ishii-VirtualBox:~$ cd /home/ishii/yasuwiki/
ishii@ishii-VirtualBox:~/yasuwiki$ head revert.dat
Portal:スポーツ/今日は何の日      3654      1114      2540
Portal:文学/今日は何の日          3922      1518      2404
朝ズバツ!                2572      668      1904
Template:今日は何の日      4470      2884      1586
Portal:災害/今日は何の日      1421      500      921
ログ・ホライズン          1312      466      846
デュラララ!!              1432      607      825
KARA              1941      1133      808
水橋かおり          1077      311      766
魔法少女まどか☆ギカ      2683      1933      750
ishii@ishii-VirtualBox:~/yasuwiki$

```

図 5.4 差し戻し回数の多いページ

5.1.5 編集者の調査

「stub-meta-history」の中から、「pageId[tab]revisionId[tab]userId[tab]ip」というデータを作る．詳細は以下の editors2.py を参照．

editors2.py

```
# -*- coding: utf-8 -*-

import xml.sax
import sys

class myHandler(xml.sax.ContentHandler):
    reading = {"id":False, "contributor":False, "revision":False, "ip":False, "timestamp":False}
    pageId = ""
    revisionId = ""
    userId = ""
    ip = ""
    timestamp = ""

    def __init__(self):
        xml.sax.ContentHandler.__init__(self)

    def startElement(self, name, attrs):
        self.reading[name] = True

    def endElement(self, name):
        self.reading[name] = False
        if name == "page":
            self.pageId = ""
        elif name == "revision":
            print(self.pageId + "\t" + self.revisionId + "\t" + self.userId + "\t" + self.ip + "\t" +
                  self.timestamp)
            self.revisionId = ""
            self.userId = ""
            self.ip = ""
            self.timestamp = ""

    def characters(self, content):
        if self.reading["id"] == True:
            if self.reading["contributor"] == True:
                self.userId = self.userId + content
            elif self.reading["revision"] == True:
                self.revisionId = self.revisionId + content
            else:
                self.pageId = self.pageId + content
        elif self.reading["ip"] == True:
            self.ip = self.ip + content

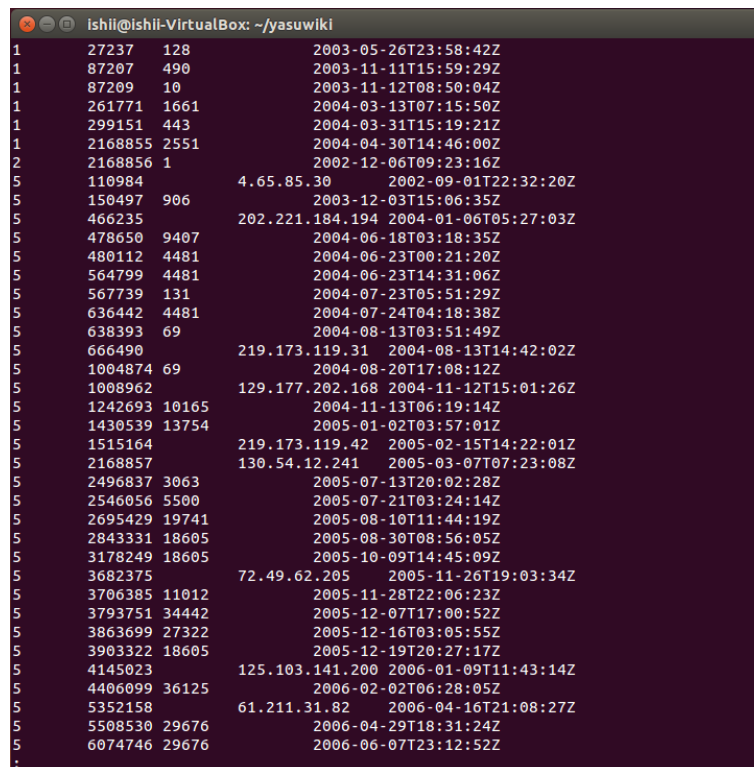
        elif self.reading["timestamp"] == True:
            self.timestamp = self.timestamp + content
```

```
def main():
    xml.sax.parse(sys.stdin, myHandler())
if __name__ == "__main__":
    main()
```

以下のコマンドを入力した。

```
cat jawiki-20150901-stub-meta-history.xml | python3 editors2.py | less
```

結果は以下のとおりである。データとしてとる場合は「| less」の部分を「> editors2.dat」とした。



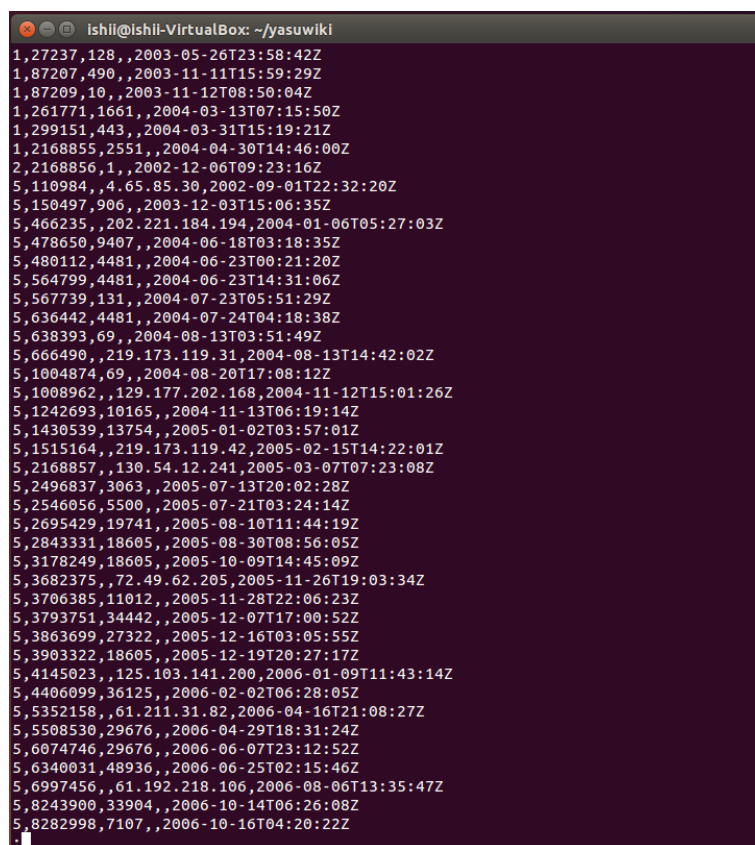
```
ishii@ishii-VirtualBox: ~/yasuwiki
1 27237 128 2003-05-26T23:58:42Z
1 87207 490 2003-11-11T15:59:29Z
1 87209 10 2003-11-12T08:50:04Z
1 261771 1661 2004-03-13T07:15:50Z
1 299151 443 2004-03-31T15:19:21Z
1 2168855 2551 2004-04-30T14:46:00Z
2 2168856 1 2002-12-06T09:23:16Z
5 110984 4.65.85.30 2002-09-01T22:32:20Z
5 150497 906 2003-12-03T15:06:35Z
5 466235 202.221.184.194 2004-01-06T05:27:03Z
5 478650 9407 2004-06-18T03:18:35Z
5 480112 4481 2004-06-23T00:21:20Z
5 564799 4481 2004-06-23T14:31:06Z
5 567739 131 2004-07-23T05:51:29Z
5 636442 4481 2004-07-24T04:18:38Z
5 638393 69 2004-08-13T03:51:49Z
5 666490 219.173.119.31 2004-08-13T14:42:02Z
5 1004874 69 2004-08-20T17:08:12Z
5 1008962 129.177.202.168 2004-11-12T15:01:26Z
5 1242693 10165 2004-11-13T06:19:14Z
5 1430539 13754 2005-01-02T03:57:01Z
5 1515164 219.173.119.42 2005-02-15T14:22:01Z
5 2168857 130.54.12.241 2005-03-07T07:23:08Z
5 2496837 3063 2005-07-13T20:02:28Z
5 2546056 5500 2005-07-21T03:24:14Z
5 2695429 19741 2005-08-10T11:44:19Z
5 2843331 18605 2005-08-30T08:56:05Z
5 3178249 18605 2005-10-09T14:45:09Z
5 3682375 72.49.62.205 2005-11-26T19:03:34Z
5 3706385 11012 2005-11-28T22:06:23Z
5 3793751 34442 2005-12-07T17:00:52Z
5 3863699 27322 2005-12-16T03:05:55Z
5 3903322 18605 2005-12-19T20:27:17Z
5 4145023 125.103.141.200 2006-01-09T11:43:14Z
5 4406099 36125 2006-02-02T06:28:05Z
5 5352158 61.211.31.82 2006-04-16T21:08:27Z
5 5508530 29676 2006-04-29T18:31:24Z
5 6074746 29676 2006-06-07T23:12:52Z
:
```

図 5.5 編集者の調査

この editors2.dat を mysql にインポートしたいため、CSV 化する。そのため、以下のようにコマンドを入力し、「,」区切りにする。csv ファイルでとるときは「| less」の部分を「> editors2.csv」とかする。

```
cat editors2.dat | awk 'BEGIN {FS="^";} {print $1","$2","$3","$4","$5}' | less
```

結果は以下のとおり、



```
Ishii@Ishii-VirtualBox: ~/yasuwiki
1,27237,128,,2003-05-26T23:58:42Z
1,87207,490,,2003-11-11T15:59:29Z
1,87209,10,,2003-11-12T08:50:04Z
1,261771,1661,,2004-03-13T07:15:50Z
1,299151,443,,2004-03-31T15:19:21Z
1,2168855,2551,,2004-04-30T14:46:00Z
2,2168856,1,,2002-12-06T09:23:16Z
5,110984,,4.65.85.30,2002-09-01T22:32:20Z
5,150497,906,,2003-12-03T15:06:35Z
5,466235,,202.221.184.194,2004-01-06T05:27:03Z
5,478650,9407,,2004-06-18T03:18:35Z
5,480112,4481,,2004-06-23T00:21:20Z
5,564799,4481,,2004-06-23T14:31:06Z
5,567739,131,,2004-07-23T05:51:29Z
5,636442,4481,,2004-07-24T04:18:38Z
5,638393,69,,2004-08-13T03:51:49Z
5,666490,,219.173.119.31,2004-08-13T14:42:02Z
5,1004874,69,,2004-08-20T17:08:12Z
5,1008962,,129.177.202.168,2004-11-12T15:01:26Z
5,1242693,10165,,2004-11-13T06:19:14Z
5,1430539,13754,,2005-01-02T03:57:01Z
5,1515164,,219.173.119.42,2005-02-15T14:22:01Z
5,2168857,,130.54.12.241,2005-03-07T07:23:08Z
5,2496837,3063,,2005-07-13T20:02:28Z
5,2546056,5500,,2005-07-21T03:24:14Z
5,2695429,19741,,2005-08-10T11:44:19Z
5,2843331,18605,,2005-08-30T08:56:05Z
5,3178249,18605,,2005-10-09T14:45:09Z
5,3682375,,72.49.62.205,2005-11-26T19:03:34Z
5,3706385,11012,,2005-11-28T22:06:23Z
5,3793751,34442,,2005-12-07T17:00:52Z
5,3863699,27322,,2005-12-16T03:05:55Z
5,3903322,18605,,2005-12-19T20:27:17Z
5,4145023,,125.103.141.200,2006-01-09T11:43:14Z
5,4406099,36125,,2006-02-02T06:28:05Z
5,5352158,,61.211.31.82,2006-04-16T21:08:27Z
5,5508530,29676,,2006-04-29T18:31:24Z
5,6074746,29676,,2006-06-07T23:12:52Z
5,6340031,48936,,2006-06-25T02:15:46Z
5,6997456,,61.192.218.106,2006-08-06T13:35:47Z
5,8243900,33904,,2006-10-14T06:26:08Z
5,8282998,7107,,2006-10-16T04:20:22Z
:
```

図 5.6 編集者の調査 csv 化

editors2.csv の userId から役割がある人（管理者とか bot とか）たちに role をつける．詳細は wikipedia が提供しているデータにある「jawiki-latest-user_groups.sql.gz」を参照する．
本研究では「jawiki-20150901-user_groups.sql.gz」を使用．中身は以下のとおり．

```

Ishii@Ishii-VirtualBox: ~/yasuwiki
-- MySQL dump 10.13  Distrib 5.5.41, for debian-linux-gnu (x86_64)
--
-- Host: 10.64.16.19    Database: jawiki
-- Server version      5.5.5-10.0.16-MariaDB-log

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `user_groups`
--

DROP TABLE IF EXISTS `user_groups`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `user_groups` (
  `ug_user` int(5) unsigned NOT NULL DEFAULT '0',
  `ug_group` varbinary(255) NOT NULL DEFAULT '',
  PRIMARY KEY (`ug_user`, `ug_group`),
  KEY `ug_group` (`ug_group`)
) ENGINE=InnoDB DEFAULT CHARSET=binary;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `user_groups`
--

/*!40000 ALTER TABLE `user_groups` DISABLE KEYS */;
INSERT INTO `user_groups` VALUES (10,'bureaucrat'),(10,'sysop'),(443,'sysop'),(4
72,'sysop'),(474,'sysop'),(1523,'sysop'),(1772,'sysop'),(1881,'abusefilter'),(18
81,'sysop'),(4054,'sysop'),(6130,'interface-editor'),(6896,'abusefilter'),(6896,
'sysop'),(7229,'sysop'),(8405,'sysop'),(9795,'sysop'),(11012,'sysop'),(11300,'sy
sop'),(11928,'abusefilter'),(11928,'sysop'),(12316,'sysop'),(14948,'abusefilter'
),(14948,'sysop'),(16156,'abusefilter'),(16760,'abusefilter'),(16760,'bureaucrat
'),(16760,'sysop'),(19084,'ipblock-exempt'),(20162,'abusefilter'),(20162,'checku
ser'),(20162,'oversight'),(20162,'sysop'),(20609,'sysop'),(20846,'bot'),(28341,'
abusefilter'),(28341,'sysop'),(31942,'bot'),(31942,'ipblock-exempt'),(32227,'sys
op'),(35218,'sysop'),(35909,'abusefilter'),(35909,'bureaucrat'),(35909,'checkuse
r'),(35909,'oversight'),(35909,'sysop'),(37161,'abusefilter'),(37161,'sysop'),(3
:

```

図 5.7 ユーザーグループ情報

5.1.6 外部結合

「editors2.csv」と「jawiki-20150901-user_groups.sql」の外部結合を行う。
外部結合を行う際は mysql 内で行う。

インポート

2 つのファイルを mysql へインポートする。

editors2.py の場合

ローカル内で mysql へ接続して行った。
まずインポート先のテーブル作成を行う。以下のように入力する。

```
mysql>CREATE TABLE editors(
    pageId INT NOT NULL,
    revisionId INT NOT NULL,
    userId VARCHAR(30) NOT NULL,
```



```
ip VARCHAR(30) NOT NULL,
timestamp TIMESTAMP NOT NULL,
KEY (userId)
)DEFAULT CHARSET=utf8;
```

そして、作成した editors テーブルへ「editors2.csv」ファイルをインポートする。以下のように入力する。
(この作業は3時間くらいかかる。)

```
mysql>LOAD DATA LOCAL INFILE "/home/ishii/yasuwiki/editors2.csv" INTO TABLE editors
FIELDS TERMINATED BY ',' ENCLOSED BY '"'
(pageId,revisionId,userId,ip,timestamp);
```

インポートが完了したら、以下のように入力し、中身を確認する (LIMIT をつけているのはデータ量が膨大なため、全て処理すると時間がかかってしまうから)

```
mysql>SELECT * FROM editors LIMIT 100;
```

結果はこうなっていれば終了。

```
mysql> SELECT * FROM editors LIMIT 100;
```

pageId	revisionId	userId	ip	timestamp
1	27237	128		2003-05-26 23:58:42
1	87207	490		2003-11-11 15:59:29
1	87209	10		2003-11-12 08:50:04
1	261771	1661		2004-03-13 07:15:50
1	299151	443		2004-03-31 15:19:21
1	2168855	2551		2004-04-30 14:46:00
2	2168856	1		2002-12-06 09:23:16
5	110984		4.65.85.30	2002-09-01 22:32:20
5	150497	906		2003-12-03 15:06:35
5	466235		202.221.184.194	2004-01-06 05:27:03
5	478650	9407		2004-06-18 03:18:35
5	480112	4481		2004-06-23 00:21:20
5	564799	4481		2004-06-23 14:31:06
5	567739	131		2004-07-23 05:51:29
5	636442	4481		2004-07-24 04:18:38
5	638393	69		2004-08-13 03:51:49
5	666490		219.173.119.31	2004-08-13 14:42:02
5	1004874	69		2004-08-20 17:08:12
5	1008962		129.177.202.168	2004-11-12 15:01:26
5	1242693	10165		2004-11-13 06:19:14
5	1430539	13754		2005-01-02 03:57:01
5	1515164		219.173.119.42	2005-02-15 14:22:01

図 5.8 editors テーブルの中身を確認

jawiki-20150901-user_groups.sql の場合

phpmyadmin からインポートを行った。

第 6 章

考察

第 7 章

結論

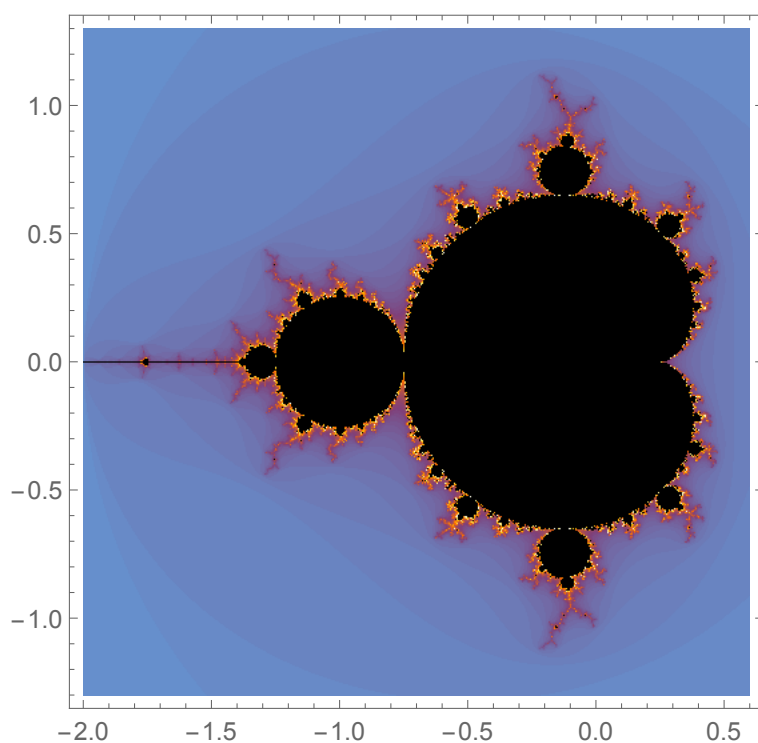


図 7.1 図の挿入例

参考文献は文献ファイル（この文書では biblio.bib）に記述し、`\cite` で参照する。例：データベースのための問い合わせ言語 SQL で数独を解く方法が提案されている [9]。このように参照すると、参考文献リストに自動的に登録される。文献の種類には、雑誌論文 [9] や会議録論文 [10]、卒業論文 [11]、書籍 [12]、ウェブサイト [13] などがある。文献の種類によって必要な項目が異なるため、biblio.bib を見て確認すること。

参考文献

- [1] アンドリュー・リー. ウィキペディア・レボリューション 世界最大の百科事典はいかにして生まれたか. 株式会社早川書房, 2009.
- [2] Wikipedia:チュートリアル 編集. https://ja.wikipedia.org/wiki/Wikipedia:%E3%83%81%E3%83%A5%E3%83%BC%E3%83%88%E3%83%AA%E3%82%A2%E3%83%AB_%E7%B7%A8%E9%9B%86 (2014.9.15 閲覧).
- [3] 清野克行. Google BigQuery ではじめる自前ビッグデータ処理入門. 株式会社秀和システム, 2014.
- [4] Project Management Institute. プロジェクトマネジメント 知識体系ガイド (PMBOK ガイド). Project Management Inst, 2009.
- [5] Wikipedia のダウンロードできるデータファイル一覧. http://www.mwsoft.jp/programming/munou/wikipedia_data_list.html (2015.05.20 閲覧).
- [6] Xml パーサ - it 用語辞典 e-words. <http://e-words.jp/w/XML%E3%83%91%E3%83%BC%E3%82%B5.html> (2015.06.01 閲覧).
- [7] Dom パーサ - it 用語辞典 e-words. <http://e-words.jp/w/DOM-1.html> (2015.06.01 閲覧).
- [8] Sax - it 用語辞典 e-words. <http://e-words.jp/w/SAX.html> (2015.06.01 閲覧).
- [9] 矢吹太郎, 佐久田博司. SQL による数独の解法とクエリオブティマイザの有効性. 日本データベース学会論文誌, Vol. 9, No. 2, pp. 13–18, 2010.
- [10] 矢吹太郎, 増永良文, 森田武史, 石田博之. 知識体系のエリア自動抽出のためのユニット分類手法. 第 5 回データ工学と情報マネジメントに関するフォーラム (DEIM2013). 電子情報通信学会データ工学研究専門委員会, 日本データベース学会, 情報処理学会データベースシステム研究会, 2013.
- [11] 久保孝樹. チケットを活用するオープンソースソフトウェア開発の実態調査. 卒業論文, 千葉工業大学, 2014.
- [12] 奥村晴彦, 黒木裕介. L^AT_EX2e 美文書作成入門. 技術評論社, 第 6 版, 2013.
- [13] 矢吹太郎. 自分のコードを出力するプログラム. <http://www.unfindable.net/article/self.html> (2012.12.01 閲覧).