Vol.2020-CH-124 No.5
2020/9/5

仏教研究におけるテキスト検索の現状と課題

永崎研宣[1]　大向一輝[2]　下田正弘[2]

概要：仏教研究において，テキスト検索は今やなくてはならない重要な位置を占めている．検索しやすく，かつ，研究者にとって見やすいテキスト検索には，Google 検索等とは異なるニーズが存在する．発表者らは，SAT 大蔵經 DB を通じて 2008 年より Web を通じてテキスト検索サービスを開発・提供してきた．本発表では，これまでのテキスト検索サービスの変遷について検討し，同 DB における今後のテキスト検索の課題を提示する．

キーワード：仏教研究，テキスト検索，異体字検索，SAT DB

The State of Text Searching in Buddhist Studies

Kiyonori Nagasaki[†1]　Ikki Ohmukai[†2]
Masahiro Shimoda[†2]

Abstract: Function of full text search has become a fundamental element in the filed of Buddhist studies. There are several requirements which are not provided by Google and other general services. We will present our challenges on it through experiences that we have been developing such services in the history of SAT Daizōkyō Database since 2008.

Keywords: Buddhist Studies, Full text search, Variant Characters, SAT Daizōkyō Database

1. はじめに

仏教研究において基本典籍の全文テキスト検索が行えるようになってすでに 10 年以上が経過している. 本発表では，仏教研究におけるテキスト検索の要件について，SAT 大蔵経データベース研究会（SAT 研究会，代表：下田正弘）[1]が 1994-2007 年に 200 人以上の人手をかけて構築した 1 億字超の仏典テキストデータを基盤とする SAT 大蔵経データベース（SAT DB, 2020 年 7 月のアクセス数は 2,187,264 件）においてこれまでに開発・提供してきた検索サービスを手がかりとして検討する．なお，二次資料としての学術論文検索も研究においてはきわめて有用ではあるものの，そのようなことであるため，ここでは研究の対象となる仏典テキストの検索を扱うものとする．

2. テキスト検索の要件

仏教研究であっても，テキスト検索においてまず求められるのは，単語やフレーズの検索である．単にそれだけでよいのであれば通常の全文テキスト検索と相違ない．しかしながら，仏教研究のなかでも特に「どのように経典を理解すべきか」ということを巡る仏教思想研究に焦点をあてるなら，船山[2]が指摘するように，さらに検討すべき要件がある．すなわち，この分野においては，文献間の思想的な影響関係を検討することが一つの研究手法となっていることから，テキスト間の関係が重要になる．そのような場合の関係としては，まず，典籍単位での関係として，経典に対する解釈を書いた注釈書，注釈書をさらに詳しく説明する再注釈書，それらの注釈書に対する批判，再批判，といったものがある．そして，一つの典籍において様々な典籍への言及や引用が行なわれることも多いことから，文章やフレーズ単位での関係が鍵となることもある．一方，そのような明示的な関係は持たないにしても，内容的に影響関係が明らかに見受けられるという場合もある．したがって，仏教研究におけるテキスト検索のニーズには，そのような関係について研究者が発見し判定できるようにするということも含まれることになる．

3. 検索対象とすべきテキスト

3.1 テキストの状況

冒頭に述べたように，今回の検索対象のテキストとなるのは仏典テキストである．研究対象となる仏典テキストの大部分は著者による自筆原稿が残されていない．それらを大別すると(1)写本・木版等で伝承されてきたテキスト，(2)伝承されてきたテキストを校合してまとめたテキスト，ということになる．（1）は，伝承の過程でのテキストの変化（修正・加筆・省略・誤写等）が生じることがあり，時期・場所によって異同が存在することが多い．そのままでは議

1 一般財団法人人文情報学研究所
International Institute for Digital Humanities
2 東京大学大学院人文社会系研究科
The University of Tokyo

論が困難なため，研究の基盤を形成すべく，(1)を比較対照し，共通のテキストとしてまとめあげるために(2)が作成される．このような状況では，共通のテキストとしての(2)が検索対象となるのが効率的だが，一方，(2)においてまとめあげられたものが必ずしも適切でないということと，むしろある時期・場所のテキストを特定して検索したいというニーズも存在するため，(1)のテキストについても選択的に検索できることが望ましい．なお，SAT DB が依拠する大正新脩大蔵経は(2)の典型である．校合の対象となった版本・写本中の異同を脚注に記しており，SAT DB はその脚注も含めた電子化を行なったため，構造化データとして異同の確認もできるようになっている．

3.2 テキストの言語

仏典テキストは，言語としては，様々な言語に翻訳されてきているが，大部になるものには，パーリ語，サンスクリット語，漢文，日本語，チベット語があり，それぞれに電子テキストがすでに多く作られている．このうち，SAT DB は漢文及び日本語の仏典テキストを電子化したものであり，したがって，ここでは漢文と日本語が混在するテキストの検索を扱う．これに加えて，テキストの一部には悉曇文字を含んでいる．

3.3 字形の違い

漢文や日本語の検索においては，文字の形の違いをどのように扱うか，という課題がある．同じ意味でも異なる字の形が用いられている場合，それが何らかの意味を持つ可能性があり，あるいは，異なる伝承の系譜にあることを示す痕跡である場合も考えられる．このようなことから，同じ意味の文字だからと字の形を統一して検索できるようにしてしまうことは必ずしも良いことではない．字の形が異なる複数の文字そのまま記述しつつそれらを横断的に検索することは現在ではそれほど難しいことではないため，あくまでも可能な限りということではあるが，字の形は元のテキストに即したものとすることを SAT DB では志向した．

3.4 外字の問題

漢文や日本語のテキストデータ，とりわけ前近代のものについては，既存の文字コードのみでは表現できない文字が含まれざるを得ず，テキスト検索を容易ならざるものとしている．SAT DB では，テキストデータの文字エンコーディングは UTF-8 とし，表現できない文字（外字）については，独自に構築した外字データベースにおいて設定した SAT 外字番号を用い，実体参照の形でテキストデータに記述するとともに，表示の際には対応する文字画像を代替表示する仕組みとなっている．また，この外字データベース中の 3000 字ほどの文字は, SAT 研究会が主体となって IRG (Ideographic Research Group) を通じて ISO/IEC10646 に符号化提案し[3]，Unicode 10.0 及び 13.0 において表現できるようになった．

4. SAT DB における実装

SAT DB における検索サービスにおける検索ソフトウェアは，SAT DB 2008 年版（SAT2008）から 2015 年版（SAT2015）までは PostgreSQL をベースにし，2018 年版(SAT2018)では Apache Solr に移行した．なお，当初 PostgreSQL を利用していたのは, 2005 年から稼働していた Web コラボレーションシステム[4]がデータの格納に PostgreSQL を採用しており，運用の互換性を考慮したためである．また，SAT2008 では，検索するごとに頁全体をリロードする形になっていたが，SAT2012 以降では，AJAX の仕組みを用い，検索時にサーバから送出するデータを検索結果データのみとした．すなわち，SAT2012 と SAT2018 で大きな変更が行なわれたことになる．以下，これについての詳細を説明する．

4.1 SAT2008 での実装

SAT2008 は SAT DB の Web 版の最初のバージョンであったが，これをリリースした時期は，1 億字のテキストデータをパソコンで簡単に検索できる時代ではなく[a]，また，ネットワークの帯域幅もそれほど十分ではなかったため，検索サービスは，サーバ上で全文検索した上で検索結果のみをクライアントに返戻するという形にすることを企図した．

データの格納に関しては, 元になった大正新脩大蔵経が，巻・頁・段・行という位置情報によって同定されるテキストであったため，テキスト検索した後に容易にこの情報を取得できるように，そして，この位置情報からテキストを得られるようにする必要があった．しかしながら，そのままで検索してしまうと行・段・頁をまたぐ単語やフレーズを検索できないため，検索用テキストと表示用テキストを別途作成することで，双方の問題を同時に解決できるようにした．

SAT DB のテキストには，行・段・頁の境界だけでなく，様々なタグが含まれており，これらも通常の全文検索ソフトではまたいで検索することが容易ではなかったため，検索用テキストからはタグも削除した．しかしながら，そのままでは，区切りがないテキストになってしまい，表示用テキストとの位置合わせがかなり大きな単位になってしまい，検索結果の確認がやや面倒になってしまう．このような場合，段落単位で位置合わせすることが一つの解決策になるが，元になった大正新脩大蔵経では段落が必ずしも妥当なものでないとされていたために，SAT DB テキスト作

[a] たとえば，当時，比較的高スペックの部類に入るノート PC であったレッツノート W7 のモデルが DRAM 最大 2GB であった．

成にあたり，段落とみなし得る改行があったとしてもそれが段落であることを明示的にデジタル化しなかった．しかしながら，段落にあたる情報がないために，巻よりも小さな単位でテキストを区切ることができず，検索テキストから表示用テキストに移行する際に適切な箇所を差し示すことが困難になるという状況であった. そこで, このときは，一行あたり字数が周囲の字数よりも明らかに少ない行を区切りとする疑似的な段落を機械的に設定し，その単位で検索を行なうことにした．すなわち，検索テキストは，疑似改行の開始行，終了行，検索テキスト，巻番号，典籍番号という構成のタブ区切りテキストとし，検索テキストの部分を全文検索するようにして，検索結果リストではこの検索テキストからスニペットを生成して表示し，そこに含んだ開始行・終了行の情報を表示用テキストとリンクすることで，対応する箇所を容易に表示できるようにした．表示用テキストは, 一行一レコードとして PostgreSQL に格納しており，リクエストに応じて対応する行を取り出して HTML に整形し，クライアントに返戻するという一連の動作がオリジナルの PHP プログラムによって実現されていた.

疑似段落単位での検索とした場合，検索語のヒット数を確認することができない．数えるためには少し時間がかかってしまい，検索のレスポンスが悪くなってしまう．そのため，ここではヒット数を算出するためだけのプログラムを作成し，部分的に AJAX を用いて，ヒット数だけは別途計算して表示する仕組みとした．この数は学術論文でも時折用いられており，ニーズが着実に存在することがうかがえる.

全文検索には，当時，Senna [5]という全文検索ライブラリが提供されており, これを PostgreSQL の任意のフィールドに対して適用するための Ludia[6]というライブラリもフリーソフトとして提供されていたため，これらを組み合わせることで全文検索を実現していた．Senna には検索インデックス作成方式として日本語形態素解析を行なう方式と文字単位での n-gram を用いる方式とが用意されていたが，ここでは，漢文のみならず日本語文についても通常の形態素解析を適用できるようなものではなかったため，n-gram によるインデックスとした．検索式は，通常の SQL クエリを多少拡張するだけで済むため，導入・開発は比較的容易であった.

異体字検索に関しては，CHISE[7]の文字同士の関係の情報を用いて，検索語に含まれる異体字を有する文字を用いて検索語候補を作成し，PostgreSQL の OR 検索を用いることでそれらをすべて検索することで対応していた．したがって，異体字を含む文字が検索語に多く含まれている場合には検索に非常に時間がかかってしまったり，検索結果が戻ってこなくなるということもあったが，体感的には 5 つ程度の異体字までは問題なく一瞬で検索結果が戻ってきていた．また，Senna では，Unicode のライブラリを介して互換漢字を正規化して検索する機能がデフォルトで提供されていたが，SAT DB では，互換漢字についても上記の仕組みで対応し，その一方で，異体字を区別して検索することもできるようにする機能も提供するために，この正規化検索機能は無効化していた[b].

なお，全文検索機能ではないが，頁・段・行を含む URL でアクセスした際にそれに対応する箇所を表示する仕組みもこのときから提供している.

4.2 SAT2012 での実装

SAT2008 では，検索ごとに頁すべてを更新する形で検索機能を提供していた．しかし，このままではアクセスごとに全ページを書き換えてしまい，Web 頁中の検索箇所以外の状態をうまく維持できず，また，検索時にやりとりされるデータ容量も大きくなってしまうという課題があった．さらに，SAT2012 では，英訳大蔵経とのパラレルコーパスや頁画像の表示機能など，頁全体の遷移が発生するとユーザビリティが下がってしまうような機能が複数実装[8]されることになったこともあり，検索の際に書き換えが必要になる箇所のみを AJAX で書き換える仕組みを採り入れることで，この課題を解決することに取り組んだ．このときには, 当時このような用途に広く用いられていた Javascript ライブラリである jQuery を利用した.

検索方法についてはこれまでと特に変わるところはなく, 検索クエリに対してサーバから返戻されたデータを指定したエレメントに表示させる，というごく一般的な仕組みとして提供した.

4.3 SAT2018 での実装

SAT2018 では，全文テキスト検索機能の部分に Apache Solr[9]を採用した．すでに SAT2015 の時点で検証実験を始めていたものをようやく実現する形となった．移行の理由を挙げればきりがないが，大きな理由のいくつかを挙げておくと，PostgreSQL の日本語全文検索ライブラリとして利用していた Senna の開発が終了したこと，RDBMS の本来の使い方をしているわけではないのでオーバーヘッドが大きくなってしまっていること, PC 等のローカルクライアントへの移植が難しいこと，等であった．また，同じ Apache Lucene を使用する ElasticSearch という選択肢もあり，若干の検証実験にも着手していたが，すでに検証実験を重ねて積み重ねが十分にある Apache Solr をこのときは採用することにした.

ここでは検索用テキストのみを Apache Solr に載せ, それ以外のデータはこれまでどおり PostgreSQL に載せて適宜引き出す形とした．ただし，Web の状況が変化してきたた

[b] なお，ある時期，設定が不十分でこの機能が有効になっていたことがあった．これはユーザからの指摘により無効化した.

め，それにあわせて枠組みの変更を行なった．大きな変化としては，クライアントや Web ブラウザが高度化し，それまでに比べて大きなデータを容易に扱えるようになったこと，サーバ・クライアント間で gzip 圧縮をかける手法が一般化したこと，が挙げられる．これにより，サーバがデータを返戻する際にあまり絞り込まずに gzip 圧縮をかけて送出し，Web ブラウザ側で表示する際に適宜絞り込みをかけるという手法が現実的なものとなった．それまではヒットした疑似段落を元に検索結果のスニペットをサーバ側で作成して返戻していたため，データ量は 3kB 程度であった．しかしながら，上記の事情から 2018 年時点ではすでにそこまでデータ量を切り詰める必要がなくなったと判断し，「ヒットした巻を丸ごと返戻して Web ブラウザ側でスニペットを作成する」という仕組みとした．これにより，同じ検索語で返戻されるデータは 466kB となっている．ただし，通信時には gzip 圧縮されているため，98.2kB である．これを Web ブラウザが受けて Javascript （主に jQuery）でスニペットを生成して表示するとともに表示用テキストへのリンクも作成するのだが，これまでに比べてかなりデータ量は大きくなっているものの，特に問題なく動作している．クライアント側で整形を行なうことになったことで，KWIC 表示の前後の文字列の長さを変更する際にサーバとのやりとりをしなくて済むようになるなど，UI 操作時のレスポンスが向上した．また，検索結果を他のサービスに援用しやすくなったという点も今後の可能性として期待したいところである．

表示用テキストについては，PostgreSQL から位置情報をキーにして引き出すことも含めてそれまでのものを踏襲している．表示用テキストの DOM（Document Object Model）を用いて，辞書検索や関連論文検索，パラレルコーパス検索，頁画像表示等の様々なサービスにデータを渡したり，イベントを発火させたりしており，表示用テキストの構造を変更すると工数が大幅に増えてしまうため，表示用テキストの構造はこのときには変更できなかった．

表示テキストと検索用テキストの対応づけに関しては，これまでのように行番号を細かい単位で持つことがなくなってしまったため，巻の冒頭からの本文の字数を取得し，その字数で位置を対応づけるようにした．稀に多少ずれてしまうことはあるものの，利用に支障のない範囲でのずれであるため，現時点では厳密には追及していない．

巻単位という，ある一定の基準での検索が可能となったため，AND 検索，OR 検索だけでなく，NOT 検索も意味を持つようになった．そこで，NOT 検索もインターフェイスとして実装した．

また，通常の検索クエリでは，ヒットした巻の数はすぐに得られるものの，ヒットした単語の数はすぐには得られない．そこで，これまでと同様に，ヒットした単語数を算出するプログラムを別途作成・サーバに設置し，通常の検索結果とは別に AJAX でその数値を問い合わせて表示する仕組みとした．

異体字検索については，Apache Solr の標準機能で提供されているフィルターとして solr.MappingCharFilterFactory クラスを利用することで実現できるが，SAT2018 開発時点ではこの機能に関する十分な確認ができていなかったため，それまでと同様に，検索クエリ生成時に異体字が存在する文字を組み合わせた文字列を一通り作成して OR 検索でつなぎ，それらをすべて検索するという手法を採った．

文字については，この直前に Unicode10.0 がリリースされ，SAT 研究会が提案した 2800 字超の文字が符号化されたため，それまで外字番号で扱っていた文字の多くを Unicode の文字に変換することができた．さらに，「記」等の一部の文字に関して，大正新脩大蔵経の字形をより忠実に反映するために IVS（Ideographic Variation Sequence）[10] を用いた．試してみた限りでは，IVS の採用により検索結果が影響されるということはないようであったが，IVS 対応フォントを用いなければ当該字形は表示されず通常の字形が表示されてしまう．利用者からの強いニーズに対応して実装したものの，効果がどれくらいあるのかということについては今後利用者調査等をしてみる必要があるだろう．

検索の仕方としては，サーバ側にラッパーを用意し，クライアントから来た検索クエリをラッパーで Apache Solr の検索クエリに整形して Apache Solr に問い合わせ，返戻された JSON データをそのまま Web クライアントに返戻する仕組みにした．Apache Solr では通常の全文検索以外に正規表現検索も利用することが可能であったため，2 つの検索方式を提供することとした．

通常の全文検索では，ラッパーでの検索クエリ生成の段階で，検索語をダブルクオーテーションで囲み，検索語と同じ文字列のみがヒットするようにした．そうしなかった場合，この設定の仕様上，検索語に含まれる文字を含むデータが多くヒットしてしまい，結果として，ユーザが探索する手間を増やしてしまう．ユーザからは検索していない文字列がヒットしたように見えてしまうため，

正規表現検索に関しては，Solr に標準で用意されているトークナイザー，solr.NGramTokenizerFactory クラスを利用することで実現できたが，対応可能な最大字数をフィールドタイプとして設定する必要があり，これを 8 文字に設定したところまではうまく動作したが，9 文字に設定したところ，インデックスの作成が終了したくなったため，上限 8 文字に設定した．これに関しては他のパラメータを調整することで増やせる可能性も考えられるので今後の課題としたい．なお，通常検索用のインデックスが 499MB であるのに対して，上限 8 文字の正規表現検索用インデックスは 9.9GB になっていた．

SAT2018 は 2019 年まで継続的に改良を続けていたが，その過程で，検索インターフェイスの改良を行なった．上

述のテキスト検索の要件において示した「本文探索」の一環として，SAT2018 では，当初より，以下の図 1 のように，曖昧フレーズ検索という位置づけでそのような本文探索機能を検索オプションに用意していた．ここでは，Solr の正規表現検索インデックスを用いているが，Web クライアントから検索クエリをサーバ側が受け取り，それを Solr への検索クエリへと変換する際に,「リクエストのあった検索語のうちどれか n 文字が異なっているもの」を正規表現で記述した文字列を作成して検索クエリに組み込むという機能であった．

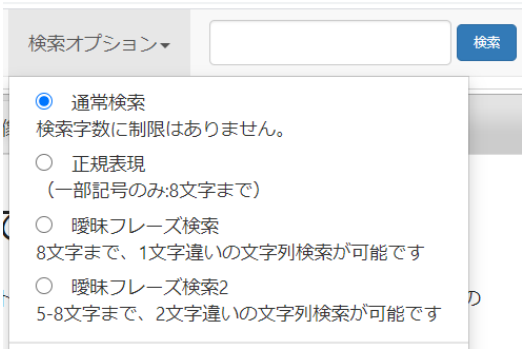

図 1　検索オプションのメニュー

しかしながら，検索オプションをわざわざ開いて確認するユーザがあまりいないためか，利用者が伸びず，2019 年度 1 年間をかけてこの機能の利用が 1097 件しかなかった．そのため，インターフェイスの改良を試みたのが，以下の図 2 である．

図 2 「本文探索」のプルダウンメニュー

このインターフェイスでは，たとえばプルダウンメニューから「本文探索:一字違い」を選択すると，あとは本文をドラッグするだけで曖昧フレーズ検索（一字違い）の検索をするようになっている．この場合，いちいち「検索オプション」を開かずとも「本文探索」ができることが，本文を読もうとするユーザからも一目瞭然である．この結果，提供を開始した 2020 年 4 月より，4/1~8/12 の約 4 ヶ月間で利用数は 2709 件となった．4 ヶ月で昨年度 1 年間の 2 倍を超えたということになる．まだ十分な告知を行なってない段階ですでにこのように利用回数が増加していることから，ある機能が提供し得る役割を，インターフェイスを通じて明確に提示することの有用性が示された形になったと言えるだろう．

5. 今後の課題

以上，SAT DB におけるテキスト検索の変遷について報告した．個別の課題としては，正規表現検索が 8 文字までしかできないという点を解消するための方策が喫緊の課題である．また，solr.MappingCharFilterFactory を用いた Apache Solr の異体字検索に一定のめどは立ったものの，それ以前の異体字検索と同様の課題，すなわち，異体字候補が多すぎると検索が止まってしまうという点については同様のようであり，パラメータ設定での対応を今一度丁寧に検討してみる必要がある．また，検索テキストと表示用テキストを分けなければならない点については，SAT DB の本文テキストを TEI/XML に置き換える作業を現在進めており，完全に XML 化できれば，XML ツールチェーンの方でよい解決策を得られる可能性がある．

一方，プロジェクトとしてスタンドアロン版の開発を進めており，これを容易に実現するためには依拠するソフトウェアをなるべく減らしてインストール・設定を簡便にする必要があり，検索閲覧システムとしての PostgreSQL を完全に離れることを目指している．データの改良・構築のためのコラボレーションシステムはデータの書き込みが行なわれるため PostgreSQL の安定性を離れることは難しいが，検索閲覧に関しては完全に Apache Solr に移行すべく，コラボレーションシステムから Apche Solr へのデータ受け渡しのフローを整備している段階である．

以上を踏まえながら現在開発中の SAT2020 では，Vue.js[11]を採用し仮想 DOM を画面描画に用いている．画面全体の描画に関しては開発の効率は高まっているように思われるが，一方で，いわゆるインラインマークアップをはじめ，仏典テキストのようなデータの場合，仮想 DOM が要求する木構造にはなじまない面があり，結果として仮想 DOM 中に HTML DOM を埋め込み，DOM が表示された後にさらにそれを操作するという形になっている．これは仮想 DOM を前提とした開発においては不便極まりないことだが，一方で，それを利用するユーザ側から見ると，仮想 DOM になじまないデータの構造がむしろ重要である，ということになる．Vue.js においても DOM の利用を制限しているわけではなく，効率は必ずしもよくないものの，利用できないわけではない．このような場合には，やはり利用者に寄り添ったデータやシステムの在り方を追求していきたい．

一方，ここまでのところ，SAT DB のすべての版は稼働

状態で運用を続けている．ユーザの中には SAT2008 や SAT2015 に愛着を持つ人も少なくなく，アクセス数を見ると未だに最もアクセス数が多いのは SAT2008 である．したがって，ユーザビリティを考慮するなら，旧サービスの維持も重要な課題である．しかしながら，旧版において当初採用したソフトウェアのほとんどすべてはバージョンアップされ，開発が止まったために別のソフトウェアに移行したものも少なくない．そのようななかで旧版を維持することは，時として新規開発に近い労力を要することもある．ソフトウェア開発においてはつきものの課題だが，この点についてもプロジェクトとユーザの実情を踏まえつつ今後検討していきたい．

謝辞　本研究は、SAT 研究会の活動に参画した多くの方々の献身により成立していることを感謝と共に記します．本研究は JSPS 科研費 JP19H00516 の助成を受けたものです．

参考文献

[1] SAT 大正新脩大藏經テキストデータベース研究会. https://21dzk.l.u-tokyo.ac.jp/SAT/ , (参照 2020-08-12).
[2] 船山 徹. 文字検索のさらなる地平に向けて ―文字列の散在的一致を網羅するために. 下田正弘・永﨑研宣編『デジタル学術空間の作り方　仏教学から提起する次世代人文学のモデル』, 2019, 文学通信. p. 169-181.
[3] 永崎研宣, 清水元広, 下田正弘. UCS 符号化提案におけるデジタルツールの活用－大正新脩大藏經外字の符号化提案にあたって. 情報処理学会研究報告, 2013-CH-97(4) (2013 年 1 月), p. 1-6.
[4] 永崎研宣, 鈴木隆泰, 下田正弘. 大正新脩大藏經テキストデータベース構築のためのコラボレーションシステムの開発. 情報処理学会研究報告, CH-70(2006 年 5 月), p. 33-40.
[5] 有限会社未来検索ブラジル. Senna 組み込み型全文検索エンジン. https://groonga.org/senna/FrontPageJ.html , (参照 2020-08-12).
[6] Ludia. https://ja.osdn.net/projects/ludia/ , (参照 2020-08-12).
[7] 文字情報サービス環境 CHISE. http://chise.org/, (参照 2020-08-12).
[8] 永崎研宣. SAT 大藏経テキストデータベース　人文学におけるオープンデータの活用に向けて. 情報管理, Vol. 58 (2015) No. 6 p. 422-437.
[9] Apache Solr . https://lucene.apache.org/solr/, (参照 2020-08-12).
[10] Ideographic Variation Database. http://unicode.org/ivd/, (参照 2020-08-12).
[11] Vue.js. https://jp.vuejs.org/index.html, (参照 2020-08-12).