The 33rd Annual Conference of the Japanese Society for Artificial Intelligence, 2019

多様なデータの統合に基づく マルチドメインナレッジベース構築システム

Multi-Domain Knowledge Base Construction System Based on Various Data Integration

山崎 朋哉 *1 Tomoya Yamazaki　真壁 拓也 *1 Takuya Makabe　西 賢太郎 *1 Kentaro Nishi　西本 智浩 *1 Chihiro Nishimoto　岩澤 宏希 *1 Hiroki Iwasawa

*1ヤフー株式会社
Yahoo Japan Corporation

Knowledge bases play crucial roles in a wide variety of information systems, such as search engines and intelligent personal assistants. For responding constantly fluctuating user information demands, we aim to construct a large-scale and well-structured multi-domain knowledge base from the world's evolving data. In this paper, we discuss enterprise-specific issues with knowledge base construction and present how to deal with these issues in our construction system. To maintain the quality of our knowledge base at the production-level, our construction system is carefully designed to incorporate various automatic and manual validation methods. We partly leverage manual validation methods to deal with business requirements and user feedback quickly since it is difficult to filter out all incorrect facts automatically in practice. Our constructed knowledge base is already utilized in real-world Japanese Web services, and the number of entities in it keeps growing steadily.

1. はじめに

近年, ナレッジベースは検索エンジンや質疑応答システムといった様々な情報システムの基盤となっている. 一般的にナレッジベースには, 主語 (Subject), 述語 (Predicate), 目的語 (Object) の 3 つ組を用いて, エンティティとそれらの関係が格納されている. 本稿では, ナレッジベースに含まれる上述した 3 つ組のことを**ファクト**と称する. 特に商用検索エンジンでは, ユーザーのクエリに対する回答を簡潔に表示するナレッジパネル *1 の表示に Google KG*2 や Satori*3 といったドメインを限定しないマルチドメインナレッジベースが活用されている. 図 1 に「ブラックジャックによろしく」のナレッジパネルと, そのパネルの情報基となるマルチドメインナレッジベース YJKB (Yahoo! JAPAN Knowledge Base) の例を示す. YJKB は本稿で提案するシステムによって構築されたマルチドメインナレッジベースで, 最も巨大な日本語のナレッジベースの 1 つである. ナレッジベース構築の手法は特定のドメインのナレッジベース構築システム [2] や, 機械学習ベースの手法 [9] など数多く提案されており, その構築方法はナレッジベースの用途によって異なる. 特に Paul ら [8] が述べているように, 実サービスに使用するシステムは, 継続的に高精度を保つ必要があり, アカデミックで提案されているシステムをそのまま導入することが困難であるケースが多い.

本稿では, 実サービスでの品質基準を満たすマルチドメインナレッジベースの構築における課題点と, その課題点に対応したナレッジベース構築システムについて提案する. 企業での品質基準を満たすナレッジベース構築の手法として, WOO [1] や Kosmix [4] が提案されている. 我々の提案するシステムは, WOO や Kosmix と同じく様々な形式の巨大なデータを取り込み, 1 つのナレッジベースを構築することを目的としているが, 以下に示す点が既存研究と比較して大きく異なる.

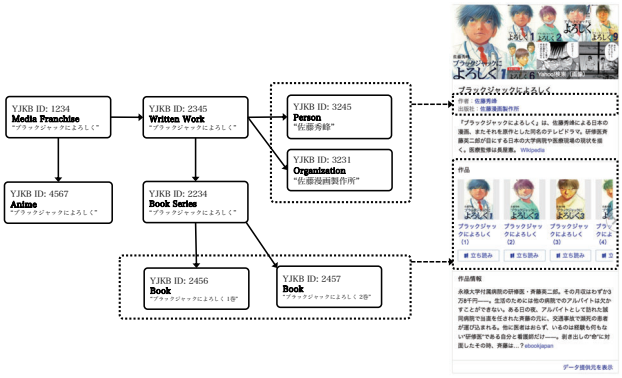

図 1: 「ブラックジャックによろしく」*4 のナレッジパネルと, そのパネルを生成するためのナレッジベース (YJKB) のイメージ図

- 様々な構造化データの情報の取り込みに加えて, Web 文書といった半構造化データから情報抽出を行う機構もシステムとして取り込んでいる点
- YJKB 中の誤った情報の修正や削除のために, オントロジ等を用いた自動の方法だけでなく, 人手で修正した情報を取り込む方法もシステムに取り込んでいる点

また, 本稿で提案するシステムを実際に日々動作して構築した YJKB は実サービスの品質基準を満たしており, 実際に日々サービスへ適用している.

本稿では, 実サービスの品質基準を満たすナレッジベース構築における課題点を挙げ, その課題を解決するシステムについて提案する. また, 実際に構築した YJKB の品質やインパクトについて説明する.

連絡先: 山崎朋哉, tomoyama@yahoo-corp.jp

*1 https://support.google.com/business/answer/6331288
*2 https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html
*3 https://blogs.bing.com/search/2013/03/21/understand-your-world-with-bing/
*4 https://search.yahoo.co.jp/search?p=ブラックジャックによろしく (最終閲覧日 : 2019 年 2 月 8 日)

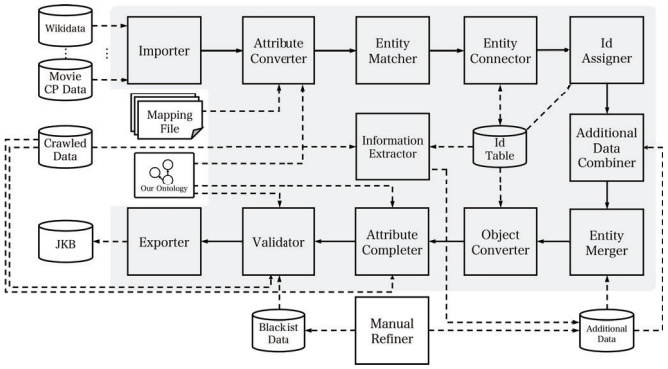

図 2: マルチドメインナレッジベース構築システム. 四角は各コンポーネントを表し, 矢印はデータの流れを表す. 点線の矢印は, システム外からのデータの移動あるいはシステム外へのデータの移動を表す.

2. マルチドメインナレッジベース構築システム

本節では, 実サービスへの品質基準を満たすシステム構築のための課題と, その課題を解決するためのナレッジベース構築システムの全体像と, 各コンポーネントの概要について説明する.

2.1 企業での品質基準を満たすための課題

ナレッジベースの構築における課題点は, 既に [1, 10] などで提案されているが, 人手の編集結果の組み込みやすさや継続的な品質維持の観点での課題点について深く言及されていない. 本稿では, 新たに 5 つの課題点を提案するとともに, 我々の提案するナレッジベース構築システムがどのようにそれらの課題点を解決しているかについて説明する.

I1: 入力データの多様性: マルチドメインのナレッジベースの構築には, 非常に多くのデータの取り込みが必要であることが知られている [3]. そのため, データフォーマットの型に依らないデータの取り込みの機構が必要である.

I2: エンティティの同一性: 同一の異なるエンティティのマッチング手法やパイプラインは [5] などで提案されている. しかし, 我々の目的は一般的な機械学習手法のように F_1 値を最大化するようなナレッジベースを構築することではなく, 精度が非常に高いことを前提とした上で, 1 エンティティの情報量を増やすことである. そのため, 高精度にエンティティをマッチングする手法が求められる.

I3: 過去のデータとの一貫性: 同じエンティティは期間を跨いで同じ ID で参照され続ける必要がある. この一貫性が保証されると, エンティティの ID を参照した編集やマッチングが可能になり, YJKB の品質向上をさらに見込むことが可能となる.

I4: 品質向上の自動化: 入力データ, 特に LOD には誤った情報が混在していることが多い. このような誤りを自動で削除や修正を行う機構や, 関連のあるエンティティ同士を自動で紐付ける機構があることで, 品質向上をスケールすることが可能となる.

I5: 人手の修正/追加情報の導入: ビジネス観点での仕様の導入や, 自動での発見が難しい誤り等を人手で修正を行うことで, YJKB を高品質に保つことが可能となる.

表 1: 入力データの比較

データソース	データ量	情報抽出の難度	更新頻度
Wikidata	30 GB	低 (JSON)	約 2 週間毎
DBpedia (日本語)	120 GB	低 (N-Triple)	約 1 年毎
Freebase	380 GB	低 (N-Triple)	2015 年 6 月 30 日に更新終了
Wikipedia (日本語)	2.6 GB	中	約 2 週間毎
社内データ	≈ 100 GB	低 (JSON, TSV, XML など)	毎日
Web からのクロールデータ	≈ 1,000 GB	高	毎日

表 2: 企業での品質基準を満たすための課題について対応を行っているコンポーネント

ナレッジベース構築の課題点	**I1** 多様性	**I2** 同一性	**I3** 一貫性	**I4** 高品質	**I5** 人手の導入
Importer	✓				
Entity Mather		✓		✓	
Entity Connector				✓	
ID Assigner		✓	✓		
Additional Data Combiner				✓	
Attribute Completer				✓	
Validator				✓	✓
Exporter				✓	✓
Information Extractor	✓				
Manual Refiner		✓			✓

2.2 システムの全体像

本稿で提案するナレッジベース構築システムの全体像を図 2 に示す. 複数の LOD (Wikidata, DBpedia, Freebase など) と様々なスキーマの社内データを Importer から Exporter までの 11 のコンポーネントを通じることで YJKB を構築する. また, クロールして得られた Web 文書から情報抽出を行う Information Extractor を組み込むことで, より多くのファクトを拡充する. 入力のデータの情報は表 1 に示す.

また, 表 2 に前節で述べた実サービスの品質基準を満たすナレッジベース構築における課題点を、どのコンポーネントで解決しているかを示す.

2.3 システムの各コンポーネントについて

Importer: 様々なスキーマのデータを YJKB のスキーマに合うようにエンティティとして取り込む. 入力のデータソースやその内容に応じた信頼度を付与することで, YJKB のエンティティ中の各ファクトは信頼度を持つ.

Attribute Converter: 各入力データのクラスと YJKB で使用する我々独自のオントロジのクラスが対応付けられたマッピングファイルを入力として与えることで, 入力データのクラスを YJKB のクラスに変換する. マッピングファイルには, 例えば Wikidata の Person クラスを表す `https://www.wikidata.org/wiki/Q215627` は YJKB のオントロジ中の PERSON クラスであるという情報が記載されている. このファイルは半自動で構築することが可能である.

Entity Matcher: 同じ対象を表すエンティティをマッチングすることで, 複数のエンティティをグループ化する. 最初に, Wikipedia URL や ISBN といった ID をベースに紐付ける. このマッチングは高精度であるが, 固有の ID を持たない多くのエンティティをマッチングさせることができない問題がある. そのため, 正規化されたエンティティの名称やエンティティのクラスなどの属性値を基にマッチングを行う手法を次に適用する. エンティティの属性の種類の集合を $\mathscr{A}$ とすると, 各エンティティは各属性毎の属性値の集合 $S = \{\{a \mid a \in A\} \mid A \in \mathscr{A}\}$ を持つ. このとき, $\mathscr{A}$ 中の全ての属性に対して, 矛盾がないエンティティ同士をマッチングさせる. この実現のために, $S = (s_1, \ldots, s_n)$ 中の各属性値集合の値の組み合わせの集合を $K = \{(k_1, \ldots, k_n) \mid k_1 \in s_1, \ldots, k_n \in s_n\}$ としたときに, 各

$k \in K$ が一致すればエンティティ同士を紐付ける．そうすることで, エンティティをノード, 紐づけを枝とみなした時に, エンティティ同士を結ぶグラフが構成できる．このグラフの内, クリーク構造を満たすエンティティ同士をグループ化することで, ID のマッチングだけでは紐付けることのできなかった多くのエンティティのグループ化が可能となる．

Entity Connector: 異なるクラスを持ち, 互いに関連するエンティティグループ同士を紐付ける．例えば, 図 1 では, **Media Franchise** のエンティティと **Written Work** のエンティティが両方とも同じ「ブラックジャックによろしく」を表したエンティティであることを判別し, それらのエンティティを関連付けることで, クラスの粒度が異なるエンティティ間の関連を表すことが可能となる．

Id Assigner: 各エンティティグループに, YJKB 用の ID を紐付ける．その際過去の ID を参照するため, 過去の YJKB と比較した場合でも同じエンティティは同じ ID を持つという一貫性を担保する．

Additional Data Combiner: 人手の修正結果や, 後述する Information Extractor (Web 文書からの情報抽出器) から取得した情報を, YJKB の各エンティティグループに紐づける．

Entity Merger: エンティティグループに含まれるエンティティを 1 つのエンティティに統合する．その際, 各エンティティの持っている信頼度が足し合わされる．この信頼度が低いエンティティは後述する Validator で除去することで, YJKB 中のエンティティの品質の高さを維持することが可能となる．

Object Converter: エンティティ中のファクトの目的語は元データソースの ID か, 文字列で表記されているため, それらを YJKB 中のエンティティの ID に解決する．元データソースの ID が入っている場合は, 元データソースと YJKB の ID の対応表を参照することができるが, 文字列で表記されている場合は, エンティティの曖昧性を解決することで YJKB の ID に解決する．

Attribute Completer: エンティティに新たな情報を追加する．新たな情報の追加の方法として, オントロジを使用する方法を提案する．例えば, YJKB には独自のオントロジを定義しており, そのオントロジでは owl の inverseOf[*5] に従った情報を追加する．これは, 欠損した情報の補完を行っているため, 推論に基づいた Knowledge Base Completion を行っていることに等しい．

Validator: エンティティ中の誤った情報の削除や修正を行う．例えば, オントロジを使用することで, 主語と述語の組み合わせがオントロジ制約違反を起こしている場合は削除するといった自動での削除機構 [7] の他, 人手で作成したブラックリストのデータに基づいて不正なデータを除去する処理を行う．また, 信頼度が閾値以下の情報は削除するロジックを入れることで, YJKB の品質の高さを担保することが可能となる．

Exporter: YJKB を使用者が扱いやすいフォーマット (JSON, N-Triple) で出力する．このとき, 使用するサービスに応じて, 権利面などの関係から使用できない情報を削除する処理も行う．

Information Extractor: クロールされた Web 文書からの情報抽出を行い, 必要な情報を YJKB に格納する．情報抽出のロジックに関する詳細は紙面の都合上省略する．

Manual Refiner: ユーザーのフィードバックや人手での品質評価によって発見した情報を基に, YJKB の品質を向上させる処理を行う．追加情報は Additional Data, 削除情報は Blacklist Data としてデータベースに格納し, それらのデータを上記コンポーネントで取り入れることで, 日々構築される YJKB に随時その結果を反映させることが可能となる．このコンポーネントを導入することで, **I5** の課題の対応を容易にしている．

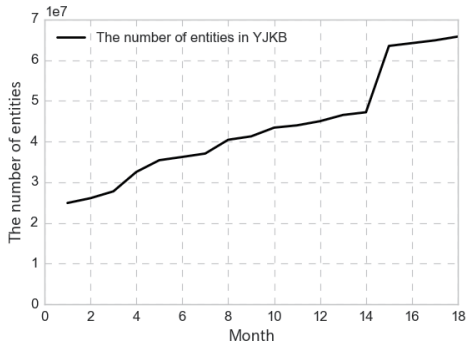

図 3: 18ヶ月間の YJKB 中のエンティティ数の推移を示すグラフ

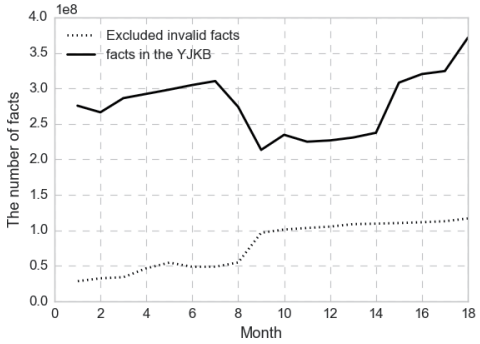

図 4: 実線は 18ヶ月間の YJKB に含まれるファクトの数, 破線は Validator 等で除去されたファクトの数の推移を示すグラフ

3. 実験結果

本稿で提案したシステムを Apache Spark で 18ヶ月に渡って日々実行した．その間のエンティティ数の変化は図 3 に示す．多様な入力データの増加とともに YJKB 中のエンティティ数も堅調に増加していることから, **I1** で示した多様性に対応できていることが分かる．

また, エンティティ数だけで無く YJKB 中に含まれるファクトの数の推移を表 5 に示す．ファクト数がエンティティ数の増加に比例していない理由は, 8ヶ月目に Validator のロジックの精度向上が行われたためである．この結果から Validator によって多くの誤りを持つファクトの自動削除が正しく行えており, **I4** への対応が実現できる．具体的に削除されたファクトの主要な条件を以下に示す．

1. オントロジのクラス制約違反 (例: Person クラスを指す述語のファクトが, Person クラス以外のクラスを目的語として持つ)
2. オントロジのプロパティ制約違反 (例: ファクトの述語が Functional[*6] の条件を満たすべきときに, その目的語がリテラル型等ではなくオブジェクト型である)

*5 https://www.w3.org/TR/owl-ref/#inverseOf-def

*6 https://www.w3.org/TR/owl2-syntax/#Functional_Data_Properties

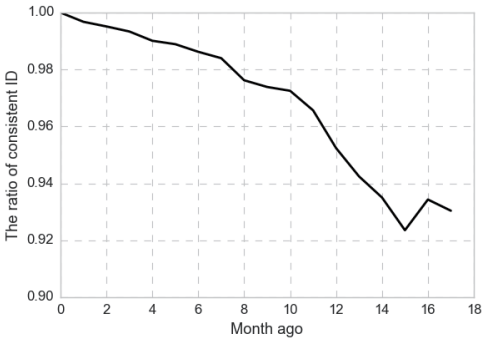

図 5: 過去データとの ID の一貫性の割合の推移を示すグラフ

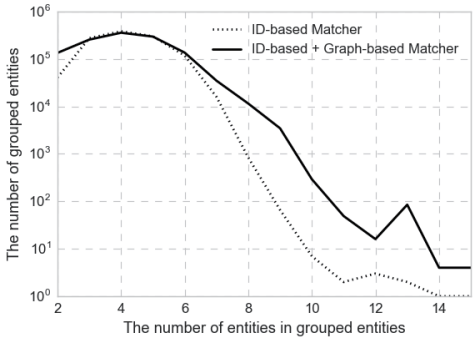

図 6: グループに含まれるエンティティの数と, そのグループの数を表す. ID の整合性によるルールベースのマッチングと, 属性値を使ったグラフベースのマッチングの 2 種類の結果を示す.

3. データタイプの不整合 (例: 目的語が URL 型であるが, 正しい URL 表記ではない場合や, ISBN が正しいフォーマットに則っていない場合)

上記の 3 条件はそれぞれ Validator で削除されたファクトの数の内, 97.7%, 2.0%, 0.3%の割合を占める. オントロジ違反を正しく検知するためには, [6] で述べられているように網羅性のあるオントロジが必要となるため, 我々が独自で定義しているオントロジが有用であることが示された.

次に, 過去のデータと比較してどの程度 ID の一貫性が保たれているかを比較する. 1 週間前のデータと比較したときに, YJKB の ID は 99.9996%の ID が一貫している. 表 5 に, 過去のデータとの YJKB の ID の一貫性の割合を比較した結果を示す. 1 年間で約 5%程度しか ID が変化しておらず, 既存研究の WOO [1] が 1 週間で約 1%弱の変化があることと比較しても ID の一貫性について頑健であることから, **I3** で示した ID の一貫性の課題への対応が実現できていることを示す.

最後にエンティティのマッチング手法の結果について比較する. 本稿では, 2.3 章で説明した通り ID を使ったルールベースのマッチング手法と属性値を使ったグラフベースのマッチング手法を適用している. 図 6 にそれぞれの手法で, 同じエンティティ群に対してどの程度マッチングするエンティティ数が変わるかを示す. ルールベースだけではマッチングすることができなかったエンティティもグラフベースの手法を導入することにより, 新たにマッチングしていることがわかる. また, グラフベースの手法は枝の張り方に制約を加えることで 99%以上の精度でのマッチングを実現する. このことから, **I2** で示したエンティティの同一性の問題の対応が実現できている..

4. まとめ

本稿では, 企業での品質基準を満たすマルチドメインナレッジベースの構築における課題点と, その課題点に対応するナレッジベース構築システムについて提案した. また, 提案したシステムを長期間動作させて構築された YJKB の品質について調査した.

ナレッジベースの欠損情報の補完, エンティティマッチング, 誤りファクトの検出の処理を機械学習等の高度な手法を企業品質基準を満たすように組み込むことが今後の課題である.

参考文献

[1] Kedar Bellare, Carlo Curino, Ashwin Machanavajihala, Peter Mika, Mandar Rahurkar, and Aamod Sane. WOO: A Scalable and Multi-tenant Platform for Continuous Knowledge Base Synthesis. VLDB 2013.

[2] Sutanay Choudhury, Khushbu Agarwal, Sumit Purohit, Baichuan Zhang, Meg Pirrung, William Smith, and Mathew Thomas. Nous: Construction and querying of dynamic knowledge graphs. In *ICDE 2017*.

[3] Nilesh Dalvi, Ashwin Machanavajjhala, and Bo Pang. An Analysis of Structured Data on the Web. In *VLDB 2012*.

[4] Omkar Deshpande, Digvijay S. Lamba, Michel Tourn, Sanjib Das, Sri Subramaniam, Anand Rajaraman, Venky Harinarayan, and AnHai Doan. Building, Maintaining, and Using Knowledge Bases: A Report from the Trenches. SIGMOD 2013.

[5] Pradap Konda, Sanjib Das, Paul Suganthan G. C., AnHai Doan, Adel Ardalan, Jeffrey R. Ballard, Han Li, Fatemah Panahi, Haojun Zhang, Jeff Naughton, Shishir Prasad, Ganesh Krishnan, Rohit Deep, and Vijay Raghavendra. Magellan: Toward Building Entity Matching Management Systems. VLDB 2016.

[6] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*, 8(3):489–508, 2017.

[7] Youen Péron, Frédéric Raimbault, Gildas Ménier, and Pierre-François Marteau. On the detection of inconsistencies in RDF data sets and their correction at ontological level. Technical report, June 2011.

[8] G.C. Paul Suganthan, Sun Chong, K. Krishna Gayatri, Zhang Haojun, Yang Frank, Rampalli Narasimhan, Prasad Shishir, Arcaute Esteban, Krishnan Ganesh, Deep Rohit, Raghavendra Vijay, and Doan AnHai. Why Big Data Industrial Systems Need Rules and What We Can Do About It. SIGMOD 2015.

[9] Sen Wu, Luke Hsiao, Xiao Cheng, Braden Hancock, Theodoros Rekatsinas, Philip Levis, and Christopher R. Fonduer: Knowledge Base Construction from Richly Formatted Data. In *SIGMOD/PODS 2018*.

[10] 市瀬 龍太郎, Kertkeidkachorn Natthawut, and 趙 麗花. 知識グラフ作成のための統合知識基盤の構築に向けて. JSAI 2018.