**DỰ ĐOÁN ÁNH XẠ LƯỢC ĐỒ VỚI CÁC BỘ DỮ LIỆU CÙNG TÍNH CHẤT BẰNG FLEXMATCHER TRONG BÀI TOÁN SO KHỚP LƯỢC ĐỒ**

**Võ Trung Nam [[1]](#endnote-1),Hoàng Thái Thanh Nhàn[[2]](#endnote-2)**

**Nguyễn Thị Bích Ngân[[3]](#endnote-3)**

**TÓM TẮT**

Bài toán so khớp lược đồ (Schema matching) là một trong những công đoạn thuộc quy trình thu thập, tổng hợp, xử lý và cấu trúc các bộ dữ liệu từ dữ liệu Web. Trong bài báo này, chúng tôi sẽ giới thiệu quy trình so khớp lược đồ sử dụng thư viện Flexmatcher[[4]](#footnote-1) của hệ thống thư viện BigGorilla[[5]](#footnote-2) và thực nghiệm dựa trên bộ dữ liệu về thông tin Sách (goodbook)[[6]](#footnote-3) được tham khảo từ trang dữ liệu Kaggle[[7]](#footnote-4).

**Từ khóa**: *so khớp lượt đồ, thư viện flexmatcher, thư viện biggorilla*

1. **GIỚI THIỆU**

Trong bài báo này, chúng tôi sẽ trình bày về khái niệm và cách thức thực hiện của bài toán so khớp lược đồ (Schema matching) và lợi ích khi áp dụng bài toán. Tiếp đến, chúng tôi sẽ giới thiệu về thư viện Flexmatcher (thư viện thực hiện ánh xạ của Big Gorrila), được dùng để thực hiện dự đoán ánh xạ của phần tử giữa các lược đồ được thu thập từ nhiều nguồn khác nhau. Đồng thời, chúng tôi sẽ đưa ra quy trình hoạt động của thư viện này. Cuối cùng là kết quả thực nghiệm của bài toán, chúng ta sẽ thu được bộ ánh xạ của dữ liệu được so khớp từ các nguồn dữ liệu khác nhau.

1. **SCHEMA MATCHING**

Schema matching là một kỹ thuật trong xử lý dữ liệu. Kỹ thuật này giúp chúng ta tìm được sự liên kết, liên quan giữa các phần tử (các thuộc tính hay các mối quan hệ) của hai lược đồ[[8]](#footnote-5). Schema matching đã được công nhận là một trong các bước cơ bản trong trình tích hợp dữ liệu (process of data integration) [1] vì thế nên kết quả của Schema matching sẽ có tác động rất lớn. Sau khi thực hiện Schema matching kết quả ta sẽ thu được một bộ các ánh xạ Bộ ánh xạ này sẽ được dùng làm dữ liệu đầu vào của Schema mapping để có thể tạo ra một lược đồ lớn hơn, một truy vấn từ nhiều nguồn dữ liệu không đồng nhất, loại bỏ thông tin trùng lặp, cũng như là hợp lý hóa các hoạt động liên quan đến các nguồn dữ liệu không đồng nhất.

Để hiểu rõ hơn, chúng tôi sẽ phân tích một ví dụ như sau: giả sử, chúng ta cần tìm kiếm thông tin của một đối tượng A, và kết quả tìm kiếm chúng ta thu được ở rất nhiều nguồn dữ liệu Web khác nhau. Tuy nhiên, thông tin của đối tượng A lại không đồng nhất về dữ liệu và có thể sai lệch, khiến chúng ta gặp khó khăn trong việc xác định dữ liệu thông tin chính xác về đối tượng A mà chúng ta mong muốn tìm kiếm. Để có thể giải quyết vấn đề này, chúng ta cần phải thống nhất các lược đồ trên thành một lược đồ tổng hợp, lược đồ này sẽ chứa những phần tử tương đương và liên quan đến các lược đồ trước khi chúng ta tổng hợp. Khi xác định được phần tử nào trong các lược đồ tương đương với phân tử nào trong lược đồ tổng hợp thì chúng ta có thể bắt đầu thể thực hiện ánh xạ. Từ đó, chúng ta thu được một lược đồ tổng hợp chứa các thông tin sau khi gộp từ các lược đồ khác nhau và dữ liệu được rút ra từ lược đồ này có thể khả quan và có độ chính xác cao hơn.

Ví dụ 1:

Chúng tôi nói về việc tìm kiếm thông tin về sách. Giả sử rằng chúng tôi sẽ tìm kiếm thông tin của một cuốn sách, và kết quả tìm kiếm thông tin của cuốn sách đó được thu thập ở nhiều trang web như Wikipedia hoặc các trang bán sách như Tiki, Fahasa, Waka, … Sau khi thực hiện các bước thu thập thông tin và chuẩn bị dữ liệu, chúng tôi thu được hai lược đồ sau đây:

* Lược đồ 1: (“Ma”, “Ten”, “DanhGia”, “MoTa”)
* Lược đồ 2: (“BookID”, “BookName”, “Rating”,” Describe”)

Từ hai lược đồ trên, chúng tôi sẽ tạo ra một lược đồ tổng hợp bao gồm các phần tử như sau:

Lược đồ tổng hợp: (“MaSach”, “TenSach”, “DanhGiaSach”, “MoTaSach”).

Bằng thủ công ta sẽ ánh xạ các lược đồ như sau:

* Lược đồ 1: (Ma = MaSach, Ten = TenSach, DanhGia= DanhGiaSach, MoTa = MoTaSach)
* Lược đồ 2: (BookID = MaSach, BookName = TenSach, Rating= DanhGiaSach, Describe = MoTaSach)

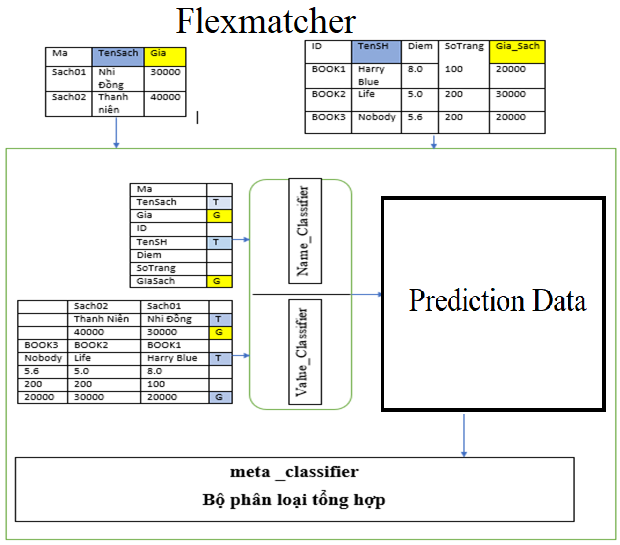
Ngoài ra, chúng tôi có thể sử dụng Lược đồ 1 ánh xạ với Lược đồ 2 vì có các phần tử có sự liên quan đến nhau. Từ đó, chúng tôi xác định được các bộ liên quan bao gồm (Ma =BookID), (Ten = BookName), (DanhGia = Rating), (MoTa = Describe).

1. **THƯ VIỆN SO KHỚP LƯỢC ĐỒ FLEXMATCHER**

Flexmatcher là một thư viện thực hiện các giải pháp nêu trên bằng ngôn ngữ Python, đồng thời cũng là một phần của dự án BigGorilla[[9]](#footnote-6) cung cấp các công cụ để tích hợp dữ liệu và chuẩn bị dữ liệu [2]. Cụ thể hơn, Flexmatcher là một hệ thống giúp chúng ta so sánh dữ liệu và tạo ra một lược đồ tổng hợp (hay còn gọi là lược đồ trung gian) từ các lược đồ của nhiều nguồn dữ liệu mà chúng ta thu thập được.

Flexmatcher sử dụng bộ các kỹ thuật máy học (machine learning) có giám sát để có thể huấn luyện được một mô hình có thể dự đoán được sự tương đương giữa các lược đồ mới so với lược đồ tổng hợp. Ý tưởng của Flexmatcher là sau khi được huấn luyện từ một bộ dữ liệu nhỏ được ánh xạ thủ công tới lược đồ tổng hợp. Sau đó, Flexmatcher có thể dự đoán được các bộ dữ liệu từ nguồn khác. Hầu hết các phần, Flexmatcher tương thích với hệ thống LSD (Learning Source Description)[[10]](#footnote-7) [3].

Trong hình 1, chúng ta có thể thấy được cấu trúc và quy trình hoạt động của Flexmatcher như sau:



***Hình 1*** *– Cấu trúc của flexmatcher*

Ví dụ trong hình 1, đầu vào là tập hợp các bộ dữ liệu về sách từ các nguồn khác nhau. Đầu tiên, chúng ta giả sử lược đồ tổng hợp sẽ bao gồm cột “Ten” quy định là màu xanh ký hiệu là **T** và “Gia” quy định là màu vàng ký hiệu là **G**. Tiếp theo, chúng ta chuẩn bị các bộ dữ liệu, như trong hình 1 mô tả, chúng ta có hai lược đồ khác nhau. Hai bộ dữ liệu này được sử dụng để như đầu vào(input) của flexmatcher. Tuy nhiên, các lược đồ này có thể khác nhau về số lượng cột cũng như là dữ liệu nhưng phải tương đương với lược đồ tổng hợp. Ở trong hình 1, sự tương đương đó được biểu hiện bằng các màu tương ứng. Ví dụ trong lược đồ bên trái, chúng ta quan sát được cột TenSach và Gia có điểm tương đương với lược đồ tổng hợp và các cột còn lại thì không.

Sau khi dữ liệu được đưa vào flexmatcher sẽ tạo ra hai bộ huấn luyện. Bộ dữ liệu đầu tiên (đặt bên trên) bao gồm tên của các cột cùng với các liên kết tương ứng. Ngược lại, ở bộ dữ liệu phía dưới bao gồm giá trị của các cột và các liên kết của chúng.

Sau đó, cả hai bộ dữ liệu được cung cấp cho các bộ phân loại. Đây là bộ phân loại cơ sở (base classifiers), bộ phân loại này sẽ phụ trách việc trích xuất các đặc trưng liên quan tới việc xây dựng mô hình dự đoán. Bộ phân loại đươc chia làm hai nhóm bao gồm phân loại theo tên cột và phân loại theo giá trị. Bộ phân loại theo tên cột, thì rất hiệu quả trong việc trích xuất các đặc trưng từ văn bản và xác định các văn bản tương đương nhau. Bộ phân loại theo giá trị trích xuất các đặc trưng từ bộ dữ liệu có thể là string, numberic và categorical hoặc trích xuất dựa trên phân phối các giá trị đầu vào như tối thiểu (min), tối đa (max), trung bình (mean),… để đưa ra dự đoán chính xác.

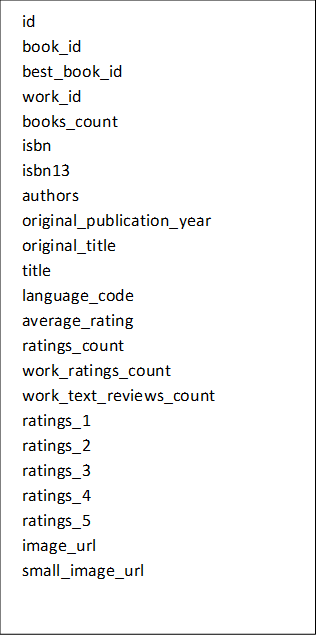
Flexmatcher sử dụng bộ phân loại cơ bản và nó vẫn cần phải xác định làm sao để kết hợp các dự đoán từ các bộ phân loại như thế nào thể trở thành một dự đoán cuối cùng. Để làm dược điều đó, đầu tiên flexmatcher sẽ sử dụng từng bộ phân loại để huấn luyện dữ liệu dự đoán. Sau só sử dụng kỹ thuật 5-fold cross validation[[11]](#footnote-8). Điều này có nghĩa là bộ dữ liệu sẽ được chia thành năm phần và mỗi phần sẽ sử dụng các phần còn lại để làm bộ huấn luyện. Sau khi thực hiện điều đó ta sẽ thu được phân phối các xác xuất trên các thuộc tính trong lược đồ tổng hợp sau đó được tổ chức thành dạng bảng và được đánh dấu như trong hình trên. Cuối cùng Flexmatcher sẽ huấn luyện bộ tổng hợp (meta-classifer) dựa trên bộ huấn luyện dự đoán đó. Bộ phân loại sẽ là một mô hình logictic regression được dùng để huấn luyện để kết hợp các dự đoán từ từng bộ phân loại cơ bản.

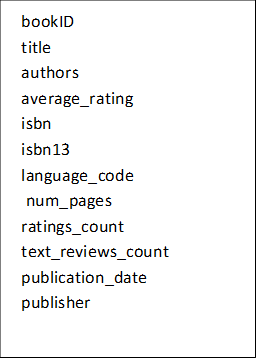
Một khi việc huấn huyện được hoàn thành. Flexmatcher có thể đưa ra dự đoán cho nguồn dữ liệu mới bằng các cách:

* + Đưa ra dự đoán dựa trên tên cột bằng bộ phân loại theo tên.
  + Đưa ra dự đoán dựa trên điểm dữ liệu thông qua bộ phân loại giá trị
  + Kết hợp tất cả dự đoán của các bộ phân loại sử dụng bộ phân loại tổng hợp đã được đào tạo. [3]

1. **KẾT QUẢ THỰC NGHIỆM**

Xét lại bài toán so khớp lược đồ về dữ liệu Sách, bây giờ chúng ta sẽ sử dụng bộ dữ liệu lớn hơn để thực hiện. Hai bộ dữ liệu ta chuẩn bị có thể không giống nhau về số lượng cột hay số lượng giá trị tuy nhiên hai bộ dữ liệu đó phải có điểm tương đương với nhau. Theo tiêu chí trên chúng tôi chọn hai bộ dữ liệu như sau



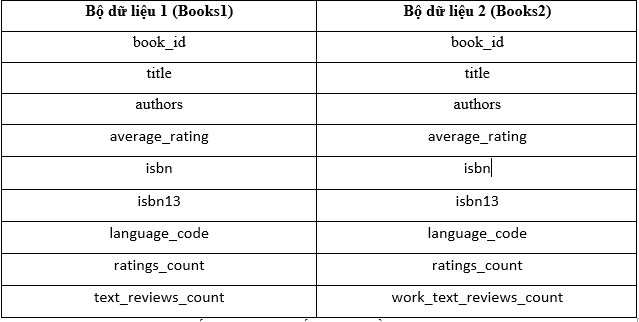


***Hình 2*** – *Bộ dữ liệu 1[[12]](#footnote-9): Books1* ***Hình 3*** – *Bộ dữ liệu 2[[13]](#footnote-10): Books2*

***Bảng 1 –*** *Thống kê bộ dữ liệu 1 và 2*

|  |  |  |
| --- | --- | --- |
|  | Bộ dữ liệu 1 (Book1s) | Bộ dữ liệu 2 (Books2) |
| Số cột | 12 | 23 |
| Số dòng | 45641 | 10000 |

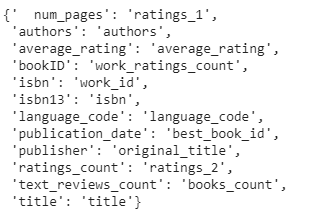
Đối với hai bộ dữ liệu 1 và bộ dữ liệu 2, ta có thể thấy được số lượng cột của bộ dữ liệu 2 nhiều hơn bộ dữ liệu 1. Vì vậy kết quả mà ta thu được có thể tùy thuộc vào bộ dữ liệu có số lượng cột thấp hơn. Tuy nhiên trước khi tới kết quả cuối cùng chúng tôi sẽ liệt kê các cặp ánh xạ mà chúng tôi mong muốn và ưu tiên những cặp có tên cột tương đương nhau mà ta có thể thấy bằng ánh xạ bằng thủ công.



***Hình 4 –*** *Bảng kết quả mong muốn ánh xạ bằng cách thủ công*

Nhưng bây giờ chúng ta sẽ sử dụng bộ dữ liệu thứ hai làm đầu vào cho flexmatcher. Nghĩa là flexmatcher sẽ sử dụng bộ dữ liệu thứ hai làm lược đồ tổng hợp. Sau đó, chúng ta sẽ dùng bộ dữ liệu tổng hợp đó để dự đoán các cặp ánh xạ của hai bộ dữ liệu.

Kết quả ta thu được như sau:



***Hình 5 –*** *Kết quả thu được từ Flexmatcher*

***Bảng 2 –*** *Thống kê sau khi thu được kết quả*

|  |  |
| --- | --- |
| *Số lượng cột ánh xạ đúng* | *4* |
| *Số lượng cột ánh xạ sai* | *8* |

Theo như hình 5, chúng ta quan sát thấy được kết quả vẫn có một vài bộ ánh xạ bị sai lệch. Đây là một thử thách để nhóm tiếp tục nghiên cứu để cải tiến, bởi vì trong dữ liệu có nhiều thông tin nhiễu. Chúng ta có thể thấy được cặp ánh xạ “isbn13” của bộ dữ liệu 1 ánh xạ tới “isbn” của bộ dữ liệu tổng hợp vì trong “isbn13” có bao gồm cả từ “isbn”. Ngoài ra, còn do cấu trúc của dữ liệu của các cột quá giống nhau nên gây ra sự sai lệch. Vì vậy, sau khi thu được kết quả trên chúng ta có thể biết được và tìm cách điều chỉnh để ít gặp nhiễu nhất có thể.

Việc này sẽ tùy thuộc vào bộ dữ liệu có cấu trúc như thế nào nên mỗi bộ dữ liệu mới sẽ phải xử lý một cách khác nhau.

1. **KẾT LUẬN**

Chúng tôi đã trình bày cơ bản về bài toán so khớp lượt đồ, cũng như là quy trình và cách hoạt động của Flexmatcher. Qua đó, những lời ích của Flexmatcher trong việc hỗ trợ giải quyết bài toán so khớp lượt đồ. Khi bài toán so khớp lượt đồ càng được cải tiến thì việc tạo ra một bộ dữ liệu lớn từ nhiều nguồn khác nhau sẽ trở nên dễ dàng hơn, từ đó tạo nên một kho dữ liệu sạch, hữu ích cho các hệ thống xử lý dữ liệu lớn khác. Mặc dù độ chính xác từ bộ dữ liệu mới có thể thấp. Flexmatcher tuy hoạt động gần đúng với kết quả mà chúng ta mong muốn. Tuy nhiên, vẫn sai khi gặp dữ liệu gây nhiễu vì vậy Flexmatcher vẫn cần bổ sung thêm các bộ phân loại khác để tối ưu hơn việc dự đoán.

1. **TÀI LIỆU THAM KHẢO**

P.A. Bernstein and S. Melnik. Metadata management. In Proceedings of the IEEE CS International Conference on Data Engineering. IEEE Computer Society, 2004

FlexMatcher Documentation: <https://flexmatcher.readthedocs.io/en/latest/>

BigGorilla: An Open-Source Ecosystem for Data Preparation and Integration

Schema Matching – Anastasios Kementsietsidis <https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-39940-9_962>

PDF - Learning Source Description for Data Integration <https://www.researchgate.net/publication/221035564_Learning_Source_Description_for_Data_Integration>

Kaggle – Bộ dữ liệu book.csv của tác giả Soumik <https://www.kaggle.com/jealousleopard/goodreadsbooks?select=books.csv>

* 1. Kaggle – Bộ dữ liệu books.csv của tác giả Foxtrot <https://www.kaggle.com/zygmunt/goodbooks-10k?select=books.csv>

**ABSTRACT**

The Schema matching problem is one of the stages in the process of collecting, synthesizing, processing and structuring data sets from Web data. In this article, we will introduce the schema matching procedure using the Flexmatcher library of BigGorilla library system and experiment based on the data set for Book information (goodbook) referenced from the Kaggle data page.

**PREDICTING THE MAPPING OF SCHEMA WITH DIFFERENT DATASETS HAVE SAME PROPERTIES BY FLEXMATCHER IN SCHEMA MATCHING**

**Vo Trung Nam1, Hoang Thai Thanh Nhan2**

**Nguyen Thi Bich Ngan3**

,2 Student of 08DHTH1 – Faculty of Information Technology, Ho Chi Minh City University Food Industry

1Email: [trungnamhufi@gmail.com](mailto:trungnamhufi@gmail.com)

2Email: [thanhnhanht219@gmail.com](mailto:thanhnhanht219@gmail.com)

3 Faculty of Information Technology, Ho Chi Minh City University Food Industry

3 Email: nganntb@hufi.edu.vn

**Keywords**: schema matching, flexmatcher, biggorilla flexmatcher.

1. ,2 SV lớp 08DHTH1 – Khoa Công nghệ thông tin, Trường ĐH Công Nghiệp Thực Phẩm TP.HCM

   Email: trungnamhufi@gmail.com [↑](#endnote-ref-1)
2. Email: thanhnhanht219@gmail.com [↑](#endnote-ref-2)
3. Khoa Công nghệ thông tin, Trường ĐH Công Nghiệp Thực Phẩm TP.HCM

   Email: nganntb@hufi.edu.vn [↑](#endnote-ref-3)
4. <https://github.com/biggorilla-gh/flexmatcher> [↑](#footnote-ref-1)
5. <https://www.biggorilla.org/> [↑](#footnote-ref-2)
6. <https://www.kaggle.com/zygmunt/goodbooks-10k?select=books.csv> [↑](#footnote-ref-3)
7. <https://www.kaggle.com/> [↑](#footnote-ref-4)
8. <https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-39940-9_962> [↑](#footnote-ref-5)
9. <https://www.biggorilla.org/> [↑](#footnote-ref-6)
10. <https://www.researchgate.net/publication/221035564_Learning_Source_Description_for_Data_Integration> [↑](#footnote-ref-7)
11. <https://medium.com/datadriveninvestor/k-fold-cross-validation-6b8518070833> [↑](#footnote-ref-8)
12. <https://www.kaggle.com/jealousleopard/goodreadsbooks?select=books.csv> [↑](#footnote-ref-9)
13. <https://www.kaggle.com/zygmunt/goodbooks-10k?select=books.csv> [↑](#footnote-ref-10)