VNUHCM - UNIVERSITY OF SCIENCE

FACULTY OF INFORMATION TECHNOLOGY

CSC14003 – Introduction to Data Science

COURSE MILESTONE 2

CLASS 23KHDL02 - GROUP 4

Students:

23127168 – Cao Trần Bá Đạt
23127238 – Trần Hoài Thiện Nhân
23127516 – Bùi Nam Việt

Lecturers:

Mrs. Nguyễn Ngọc Thảo
Mr. Huỳnh Lâm Hải Đăng
Mr. Nguyễn Thanh Tình

January 8th, 2025

Acknowledgments

We would like to express our sincere gratitude to the lecturers who provided valuable guidance and support throughout this milestone. We would especially like to thank the main lecturer, Ms. Nguyen Ngoc Thao, along with the project supervisors, Mr. Huynh Lam Hai Dang and Mr. Nguyen Thanh Tinh, for their continuous guidance, constructive feedback, and clear direction during the development of this project.

This project was conducted with the support of high-quality academic data sources, including publicly available research papers and reference datasets. We also acknowledge the contributions of the open-source community, whose libraries and tools for machine learning, natural language processing, data analysis, and text similarity played an essential role in the implementation of this work.

In addition, we appreciate the technical resources and documentation provided by the LaTeX community and prior research in reference matching and entity resolution, which significantly informed our methodological approach. This work would not have been possible without the strong open-source ecosystem and the availability of reliable academic resources.

Contents

1 Introduction

This report presents the implementation of a reference matching pipeline designed to align BibTeX entries from standardized bibliography files with reference data scraped from Semantic Scholar. The proposed pipeline addresses the challenge of identifying equivalent references across heterogeneous data sources by integrating hierarchical parsing, data standardization, machine learning-based matching, and systematic evaluation.

The system is capable of processing multi-file LaTeX documents, extracting structured BibTeX entries, and comparing them against Semantic Scholar references that may differ in format, completeness, and representation. By formulating reference matching as a binary classification and ranking problem, the pipeline enables accurate and scalable identification of corresponding references in large academic datasets.

Overall, this work aims to provide an automated and robust solution for reference integration and validation, supporting bibliographic analysis, data consistency, and large-scale academic document processing.

Group demo video link: `https://youtu.be/GrVqvtbo3WU`

2 Hierarchical Parsing

2.1 File Structure Detection

The system employs a multi-strategy approach to detect and handle LaTeX file structures.

2.1.1 Main File Detection

The main LaTeX file is identified using the following strategies:

- **Strategy 1**: Check for common main file names such as `main.tex`, `paper.tex`, `article.tex`, `manuscript.tex`, `thesis.tex`, `document.tex`, and `root.tex`.
- **Strategy 2**: Identify candidate main LaTeX files by detecting the presence of core document-level commands, including `\documentclass`, `\begin{document}`, and author declarations.
- **Strategy 3**: If only one `.tex` file exists at the root level, it is automatically selected as the main file.

2.1.2 Multi-File Input Handling

The `LaTeXFileGatherer` class manages multi-file LaTeX projects through the following mechanisms:

1. **Recursive File Gathering**: Starting from the main file, the system recursively processes all `\input{}` and `\include{}` commands.
2. **Path Resolution**:
    - First attempts to resolve paths relative to the current file's directory.

- Falls back to base directory resolution if necessary.
- Automatically appends the `.tex` extension if it is missing.

3. **Circular Dependency Prevention**: A set of processed files is maintained to prevent infinite inclusion loops.
4. **File Tracking**: The system preserves file order and tracks which files are part of the compilation path versus unused files.

2.1.3 Implementation Details

The following regular expression is used to detect include commands while ignoring commented lines:

```
PATTERN = re.compile(
    r'(?m)^(?![^%\n]*%).*\\(?:input|include)\{([^}]+)\}'
)
```

2.2 LaTeX Hierarchy Parsing

The `HierarchyParser` class constructs a hierarchical tree structure that represents the organization of the document.

2.2.1 Section Detection

The parser recognizes hierarchical sectioning commands, including:

- `\chapter`
- `\section`
- `\subsection`
- `\subsubsection`
- `\paragraph`
- `\subparagraph`

Both starred (e.g., `\section*`) and unstarred variants are supported. Hierarchy levels are maintained to ensure proper nesting of sections.

2.2.2 Special Content Extraction

The system extracts and processes special content types as follows:

- **Abstract**: Extracted using patterns such as `\begin{abstract}...\end{abstract}` or `\abstract{...}`.
- **Acknowledgments**: Detected and extracted as a separate section.
- **Block Math**: Various math environments (`equation`, `align`, `gather`, `multline`, etc.) are normalized into a standard representation.

- **Figures and Tables**: Identified through the `figure` and `table` environments.
- **References Section**: Excluded from main content parsing using multiple detection patterns.

2.2.3 Node Structure

Each node in the hierarchical tree contains the following attributes:

- `node_type`: Type of node (e.g., document, section, sentence, formula).
- `title`: Section title or content descriptor.
- `content`: Textual content of the node.
- `children`: Nested child nodes.
- `unique_id`: Content-based hash used for deduplication.
- `source_file`: Source file from which the node originates.

2.2.4 Hierarchy Construction Logic

The hierarchical tree is constructed using the following steps:

1. Parse section markers and record their positions.
2. Sort sections based on position and hierarchy level.
3. Build the tree structure while maintaining correct parent–child relationships.
4. Extract leaf nodes such as sentences and formulas within each section.
5. Assign unique identifiers to nodes using content-based hashing.

3 Standardization

3.1 Data Cleaning

The `LaTeXCleaner` class performs comprehensive text normalization to ensure consistency and remove non-semantic elements from the source documents.

3.1.1 Comment Removal

The system removes LaTeX comments while preserving meaningful content:

- Removes comments starting with the `%` character.
- Preserves escaped percent symbols (`\%`).
- Correctly handles comments that appear within math environments.

3.1.2 Formatting Command Removal

Formatting commands that do not carry semantic meaning are removed, including:

- **Spacing commands**: `\smallskip`, `\medskip`, `\bigskip`, `\hspace`, `\vspace`.
- **Page break commands**: `\newpage`, `\clearpage`, `\pagebreak`.
- **Alignment commands**: `\centering`, `\raggedright`, `\raggedleft`.
- **Font commands**: `\textbf`, `\textit`, `\emph`, `\underline`, `\texttt`.
- **Table formatting commands**: `\toprule`, `\midrule`, `\bottomrule`, `\cmidrule`.

3.1.3 Whitespace Normalization

Whitespace is normalized using the following rules:

- Multiple consecutive spaces are collapsed into a single space.
- Multiple consecutive newlines are normalized to double newlines.
- Leading and trailing whitespace is trimmed.

3.2 Math Normalization

Mathematical expressions are normalized to ensure a consistent representation across different LaTeX styles.

3.2.1 Inline Math Standardization

Inline math expressions are standardized as follows:

- `\(...\)` is converted to `$...$`.
- `\begin{math}...\end{math}` is converted to `$...$`.

3.2.2 Block Math Standardization

Block-level mathematical expressions are normalized using a unified environment:

- `$$...$$` is converted to `\begin{equation}...\end{equation}`.
- `\[...\]` is converted to `\begin{equation}...\end{equation}`.
- All block math environments such as `align`, `gather`, `multline`, `eqnarray`, and others are normalized to the `equation` environment.

This normalization ensures a consistent representation of mathematical content across heterogeneous LaTeX documents.

3.3 Reference Deduplication

The `Deduplicator` class implements a robust reference deduplication strategy to eliminate redundant BibTeX entries.

3.3.1 Content-Based Hashing

To support deduplication across document versions and reference sources, the system employs a content-based hashing strategy to generate stable and unique identifiers for hierarchical document nodes.

Each BibTEX entry generates a hash based on the following prioritized fields:

1. **Title**: The primary identifier, normalized and lowercased.
2. **Author**: The last name of the first author.
3. **Year**: Year of publication.
4. **DOI**: A highly reliable unique identifier, when available.

3.3.2 Deduplication Process

The deduplication procedure consists of the following steps:

1. **Hash Generation**: Generate a content-based hash for each entry.
2. **Collision Detection**: Identify entries with identical hashes.
3. **Title Similarity Verification**: For colliding hashes, verify that titles are genuinely similar using Jaccard similarity with a threshold of 0.6.
4. **Safe Merging**: Merge entries only if:
    - Hashes are identical, and
    - Titles are sufficiently similar under fuzzy matching.
5. **Field Merging**: When merging entries, preserve all fields from both entries using a union-based strategy.

3.3.3 Safety Mechanisms

Several safeguards are employed to prevent incorrect merges:

- Entries are kept separate if hash collisions occur but titles differ significantly.
- Entries lacking sufficient identifying fields (e.g., missing title, author, or DOI) fall back to key-based hashing.
- Original BibTEX entries are never mutated; merged entries are created as new objects.

3.3.4 Deduplication Statistics

Empirical observations show that:

- The number of original references varies across documents.
- Deduplication typically reduces the reference count by approximately 5–15%.

4 Machine Learning Pipeline

4.1 Feature Engineering

The feature extraction process (`extract_features`) generates a total of 13 features for each BibTEX–reference pair.

4.1.1 Title Features

Five features are derived from the reference title, which serves as the most discriminative signal:

1. `title_exact_match`: Binary indicator (0/1) for an exact match between normalized titles.
2. `title_ratio`: SequenceMatcher similarity ratio in the range $[0, 1]$.
3. `title_levenshtein`: Normalized Levenshtein distance similarity score.
4. `title_jaccard`: Jaccard similarity computed over sets of title words.
5. `title_word_overlap`: Proportion of overlapping words between titles.

4.1.2 Author Features

Author information is used to disambiguate references with similar titles:

6. `first_author_lastname_match`: Binary indicator for matching first author last names.
7. `author_jaccard`: Jaccard similarity over sets of authors.

4.1.3 Temporal Feature

Publication year consistency is captured using a Gaussian similarity function:

8. `year_similarity`: Defined as

$$\exp\left(-\frac{(\Delta y)^2}{2}\right),$$

where Δy is the difference between publication years.

4.1.4 Semantic Feature

Semantic similarity beyond surface text matching is incorporated using sentence embeddings:

9. `title_embedding_sim`: Cosine similarity between sentence transformer embeddings generated by the `all-MiniLM-L6-v2` model.

4.1.5 Composite Features

Multiple signals are combined to form robust composite features:

10. `title_author_score`: Average of `title_ratio` and `author_jaccard`.
11. `composite_score`: Weighted combination of multiple features:
    - Title similarity: 40%
    - Author similarity: 30%
    - Year similarity: 20%
    - First-author match: 10%

4.1.6 Feature Justification

The selected features are motivated as follows:

- **Title features** provide the strongest discriminative power and capture multiple aspects of textual similarity.
- **Author features** assist in resolving ambiguity when titles are similar.
- **Year similarity** filters temporally inconsistent matches.
- **Embedding similarity** captures semantic equivalence beyond exact lexical overlap.
- **Composite features** integrate heterogeneous signals for more robust matching.

4.2 Data Labeling

4.2.1 Manual Labeling

A small set of papers was manually labeled to provide high-quality ground truth:

- 5 papers manually annotated.
- 118 total reference matches.
- Label format:

$$\{\texttt{paper\_id} : \{\texttt{bib\_key} : \texttt{arxiv\_id}\}\}.$$

4.2.2 Automatic Labeling

Large-scale automatic labeling was performed using rule-based heuristics:

- 1,500 papers automatically labeled.
- 28,822 automatic reference matches generated.
- Matching criteria:
  - Composite score ≥ 0.5
  - Title ratio ≥ 0.3

4.2.3 Training Data Generation

The final training dataset was constructed as follows:

- For each positive match, two negative samples were generated (`negative_ratio` = 2.0).
- Negative samples consist of incorrect BibTEX–reference pairs drawn from the same paper.
- Total training examples: 80,906
  - 26,969 positive examples
  - 53,937 negative examples

4.3 Model Selection and Training

4.3.1 Evaluated Models

Three supervised classification models were evaluated:

1. **Random Forest Classifier**: 100 estimators, `random_state` = 42.
2. **Gradient Boosting Classifier**: 100 estimators, `random_state` = 42.
3. **Logistic Regression**: `max_iter` = 1000, `random_state` = 42.

4.3.2 Training Procedure

The training process follows standard supervised learning practices:

- Feature vectors are standardized using `StandardScaler`.
- All models are trained on the full training dataset.
- Training accuracy is used for initial model comparison.

4.3.3 Model Selection Results

Training accuracy for each model is reported below:

- Random Forest: 98.08%
- Gradient Boosting: 97.29%
- Logistic Regression: 95.83%

Selected Model: Random Forest Classifier, due to the highest observed training accuracy.

4.4 Data Split Strategy

The dataset is partitioned at the paper level to avoid data leakage.

4.4.1 Test Set

- Manual paper: `2304-14610` (20 labels)
- Automatic paper: `2304-14608`

4.4.2 Validation Set

- Manual paper: `2304-14656` (32 labels)
- Automatic paper: `2304-14611`

4.4.3 Training Set

- 3 remaining manual papers (66 labels)
- 1,498 remaining automatically labeled papers

4.5 Evaluation Results

4.5.1 Mean Reciprocal Rank (MRR)

Mean Reciprocal Rank (MRR) is used to evaluate ranking quality. For each BibTEX entry, the following procedure is applied:

1. Score all candidate references using `predict_proba`.
2. Rank candidates in descending order of predicted probability.
3. Identify the rank of the correct match within the top-5 predictions.
4. Compute reciprocal rank as $1/r$ if found, or 0 otherwise.
5. Average reciprocal ranks across all entries.

4.5.2 Test Set Results

- Paper `2304-14610`: MRR = 1.0000
- Paper `2304-14608`: MRR = 0.9500

Overall Test MRR: 0.9750.

4.5.3 Result Analysis

The high MRR score indicates strong ranking performance:

- Correct references are ranked first in the majority of cases.
- Perfect matching is achieved for paper `2304-14610`.
- For paper `2304-14608`, correct matches typically appear within the top two ranks.
- The combined feature set provides strong discriminative power for reference matching.

4.5.4 Potential Limitations

Despite strong results, several limitations remain:

- A small number of challenging cases prevents perfect MRR.
- Validation set evaluation is required to assess generalization.
- Performance may vary across different citation styles and domains.

5 Statistics

5.1 Parsing Success Rate

5.1.1 Data Loading

The dataset loading statistics are summarized as follows:

- BibTEX files loaded: **6,456 papers**
- Reference files loaded: **12,785 papers**
- Papers containing both BibTEX and reference data: **2,104 papers**

This corresponds to a **parsing success rate of 99.86%**, computed as

$$\frac{2{,}104}{2{,}107}.$$

5.1.2 BibTEX Parsing

A custom BibTEX parser is implemented to robustly handle real-world citation data:

- Nested braces within field values.
- Both `{value}` and `"value"` field formats.
- Multiple BibTEX entry types.
- Missing or malformed entries, which are skipped gracefully.

5.1.3 Reference Parsing

Reference metadata is parsed from JSON files with the following properties:

- UTF-8 with BOM (`utf-8-sig`) encoding support.
- Compatibility with heterogeneous reference formats from Semantic Scholar.
- Robust error handling for corrupted or missing files.

5.2 Matching Success Rate

5.2.1 Label Generation

The labeled dataset consists of both manual and automatically generated matches:

- **Manual labels**: 118 matches across 5 papers.
- **Automatic labels**: 28,822 matches across 1,500 papers.
- **Total labeled pairs**: 28,940 matches.

5.2.2 Automatic Matching Statistics

For automatically labeled papers:

- Papers processed: 1,500.
- Average matches per paper: 19.2.
- Match rate varies depending on reference density and citation style.

5.2.3 Model Performance Summary

The trained model demonstrates strong matching performance:

- **Test MRR**: 0.9750, indicating high-quality ranking.
- **Training accuracy**: 98.08% (Random Forest classifier).
- Most correct matches are ranked at position 1, with a small number appearing at rank 2.

5.3 Data Statistics Summary

Table 1 summarizes the key dataset statistics.

Table 1: Summary of dataset and model statistics

Metric	Value
Total papers processed	2,104
Papers with BibTeX	2,107
Papers with references	2,893
Manual labels	118 (5 papers)
Automatic labels	28,822 (1,500 papers)
Training examples	80,906
Test set size	Variable (2 papers)
Test MRR	0.9750
Training accuracy	98.08%

5.4 Feature Statistics

5.4.1 Feature Distribution

The extracted features exhibit the following statistical properties:

- All scalar features are normalized to the $[0, 1]$ range.
- Sentence embedding similarity is represented as cosine similarity.
- Feature standardization is applied using `StandardScaler` prior to model training.

5.4.2 Feature Importance

Feature importance analysis from the Random Forest model indicates:

- **Title-based features** contribute the highest importance.
- **Author-based features** provide medium-level importance.
- **Year similarity** has relatively lower importance.
- **Embedding similarity** serves as a complementary semantic signal.

6 Conclusion

This work presents a complete and effective reference matching pipeline with the following key contributions:

1. **Robust Hierarchical Parsing**: Support for multi-file LaTeX projects through recursive file gathering and accurate hierarchy construction.
2. **Comprehensive Standardization**: Thorough cleaning of LaTeX content, normalization of mathematical expressions, and safe reference deduplication via content-based hashing.
3. **Effective Machine Learning Pipeline**: Thirteen carefully engineered features combined with a Random Forest classifier achieve a high test MRR of 0.9750.
4. **High Success Rates**: A parsing success rate of 99.86% and consistently strong matching performance across the test set.

Overall, the system demonstrates strong effectiveness for automated reference matching. Future work will focus on validation over larger and more diverse datasets, as well as exploring improved generalization across citation styles and scientific domains.

7 References

[1] sbert.net, "SentenceTransformers Documentation", https://sbert.net/

[2] Edward Blurock, "String Similarity Metrics: Sequence Based", https://www.baeldung.com/cs/string-similarity-sequence-based

[3] evidentlyAI, "Mean Reciprocal Rank (MRR) explained", https://www.evidentlyai.com/ranking-metrics/mean-reciprocal-rank-mrr

[4] overleaf, "Learn LaTeX in 30 minutes", https://www.overleaf.com/learn/latex/Learn_LaTeX_in_30_minutes

[5] scikit-learn, "GradientBoostingClassifier", https://scikitlearn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

[6] Geeksforgeeks, "Normalizing Textual Data with Python", https://www.geeksforgeeks.org/python/normalizing-textual-data-with-python/

[7] Adrian Evensen, "Entity Resolution — An Introduction", https://medium.com/@adev94/entity-resolution-an-introduction-fb2394d9a04e