WiClean Project Report

Introduction:

The aim of this project, called WiClean, is to detect inconsistencies in Wikidata, a free and open knowledge base that acts as a central storage for structured data. Specifically, we focus on identifying cases where a person is listed as having a spouse who died before the current date. Such inconsistencies can arise due to various reasons, including data entry errors or outdated information.

To address this problem, we propose a machine learning-based solution that leverages features extracted from Wikidata and employs several classification algorithms to detect inconsistencies. The project involves querying Wikidata, preprocessing the data, feature engineering, and training and evaluating different classifiers to identify the best-performing model.

Our evaluation highlights show that the Random Forest classifier achieves the highest mean cross-validation score of 0.5074, outperforming Decision Trees and Support Vector Machines (SVMs). When tested on a holdout set, the Random Forest model achieves an accuracy of 0.522, precision of 0.527, recall of 0.894, and an F1 score of 0.663.

While our solution shows promising results, it is important to note that the performance metrics indicate room for improvement. The relatively low precision suggests that the model may be prone to false positives, identifying consistent instances as inconsistent. On the other hand, the high recall indicates that the model is effective in identifying most of the truly inconsistent instances.

Background:

Wikidata is a collaboratively edited knowledge base hosted by Wikimedia. The data in Wikidata is organized as statements about entities, where each statement consists of a property (e.g., "spouse") and a value (e.g., the identifier of a person).

Ensuring data quality and consistency is a crucial challenge in large-scale knowledge bases like Wikidata. Inconsistencies can arise due to various reasons, such as data entry errors, conflicting information from different sources, or outdated information that has not been updated. Detecting and resolving these inconsistencies is essential for maintaining the integrity and reliability of the knowledge base.

Traditional approaches to inconsistency detection often rely on rule-based systems or manual inspection. Rule-based systems define a set of predefined rules or constraints that the data must adhere to, and any violations of these rules are flagged as inconsistencies. However, creating and maintaining a comprehensive set of rules can be time-consuming and may not capture all possible inconsistencies. Manual inspection, on the other hand, involves human experts reviewing the data and identifying inconsistencies based on their domain knowledge. While this approach can be effective, it is not scalable to large datasets and can be prone to human error.

Machine learning techniques offer a promising alternative for detecting inconsistencies in large-scale knowledge bases. By training models on historical data and incorporating various features, machine learning algorithms can learn patterns and relationships that are indicative of inconsistencies. These models can then be applied to new data to automatically identify potential inconsistencies, without relying on predefined rules or manual inspection.

Data Used:

For this project, we query Wikidata to retrieve information about persons, their spouses, and the dates of death of the spouses. The query retrieves up to 100,000 results and includes the following properties:

- P26: "spouse" property
- P570: "date of death" property

The motivation for selecting this dataset is to identify cases where a person is listed as having a spouse who died before the current date, which would be an inconsistency.

It is important to note that the dataset retrieved from Wikidata may contain inherent biases or limitations, as it is a crowdsourced knowledge base. The information in Wikidata is contributed by a community of users, and the quality and completeness of the data may vary across different entities and properties. Additionally, the dataset we used is limited to 100,000 entities due to our computation constraints.

Furthermore, the dataset focuses specifically on inconsistencies related to spouses and dates of death. While this is a relevant and important type of inconsistency, it represents only a subset of the possible inconsistencies that can exist in Wikidata. Other types of inconsistencies, such as conflicting occupations, incorrect birth dates, or inconsistent relationships between entities, are not captured in this dataset.

We tried to get around the limitation of expanding for more inconsistencies, but were unsuccessful. This is because each query and even article can have a wide range of feature variables which can be hard to detect manually. We have come to the conclusion that manually choosing feature variables isn't scalable, and machine learning techniques will have to be applied to identify reasonable feature variables. Unfortunately, due to our limited knowledge in machine learning, we did not figure out how to do this.

Despite these limitations, the selected dataset provides a suitable starting point for exploring inconsistency detection in Wikidata using machine learning techniques. The focused nature of the dataset allows us to develop and evaluate our approach within the scope of this project. However, it is important to acknowledge the potential biases and limitations of the data and consider them when interpreting the results and assessing the generalizability of the solution.

Motivation:

The primary motivation behind this project is to improve the quality and consistency of data in Wikidata by automatically detecting inconsistencies. Wikidata serves as a central repository of structured knowledge that is widely used by various applications, including search engines, virtual assistants, and data analysis tools. Ensuring the accuracy and reliability of the information stored in Wikidata is crucial for these downstream applications.

The presence of inconsistencies in Wikidata can have several negative consequences. It can lead to incorrect or misleading information being propagated to downstream applications, which can impact the quality of search results, recommendations, or data-driven insights. Moreover, inconsistencies can undermine the trustworthiness and credibility of Wikidata as a reliable source of knowledge.

Traditional approaches to inconsistency detection, such as rule-based systems or manual inspection, have limitations in terms of scalability and comprehensiveness. With the increasing size and complexity of Wikidata, there is a need for automated and efficient methods to identify and resolve inconsistencies.

Our project aims to address this need by leveraging machine learning techniques to automatically detect inconsistencies in Wikidata. By training models on historical data and incorporating various features, we can learn patterns and relationships that are indicative of inconsistencies. This data-driven approach has the potential to identify inconsistencies that may be missed by rule-based systems or manual inspection.

Furthermore, our project focuses specifically on inconsistencies related to spouses and dates of death. This type of inconsistency is particularly relevant because it involves temporal information and relationships between entities. Identifying cases where a person is listed as having a spouse who died before the current date can help improve the accuracy and coherence of the knowledge base.

Design:

The solution consists of the following main components:

1. Data Retrieval: We use the SPARQLWrapper library to query Wikidata and retrieve the relevant data about persons, their spouses, and the dates of death of the spouses. The query is designed to retrieve up to 100,000 results, including the necessary properties (P26 for spouse and P570 for date of death).
2. Data Preprocessing: The retrieved data is preprocessed to extract relevant features and prepare it for machine learning. This involves several steps:
    - Extracting person and spouse identifiers: The person and spouse identifiers are extracted from the URI values in the query results. These identifiers serve as unique references to the entities in Wikidata.
    - Extracting dates of death: The dates of death of the spouses are extracted from the corresponding URI values. These dates are necessary for determining the temporal consistency of the spouse relationships.
    - Handling missing or invalid dates: In cases where the date of death is missing or invalid, appropriate strategies are applied. Missing dates are replaced with a special value (e.g., NaN), and invalid dates are handled accordingly.
    - Calculating days since death: To capture the temporal aspect of the inconsistencies, we calculate the number of days between the current date and the date of death of each spouse. This feature provides a numerical representation of the time elapsed since the spouse's death.
    - Introducing random inconsistencies: To simulate real-world scenarios and evaluate the effectiveness of our approach, we introduce random inconsistencies into a portion of the dataset. This is done by randomly selecting a subset of instances and modifying their labels to indicate inconsistency.
3. Feature Engineering: We perform feature engineering to create a suitable input for the machine learning models. This involves:
    - One-hot encoding of categorical features: The person and spouse identifiers are categorical variables that need to be converted into a numerical representation suitable for machine learning algorithms. We apply one-hot encoding to transform these identifiers into binary vectors, where each unique identifier is represented by a separate column.
    - Concatenating encoded features with numerical features: The one-hot encoded person and spouse identifiers are concatenated with the numerical feature representing the days since the spouse's death. This creates a combined feature matrix that serves as the input to the machine learning models.
4. Model Training and Evaluation: We train and evaluate three different classifiers: Decision Tree, Random Forest, and Support Vector Machine (SVM). The choice of these classifiers is based on their ability to handle both categorical and numerical features and their proven effectiveness in various classification tasks.
    - Cross-validation: We employ stratified 5-fold cross-validation to estimate the performance of each classifier. This involves splitting the dataset into five equal-sized folds, where each fold is used as a validation set while the remaining folds are used for training. The

cross-validation scores provide an estimate of how well the models generalize to unseen data.
    - Model selection: Based on the cross-validation scores, we select the best-performing model among the three classifiers. In our case, the Random Forest classifier emerges as the top performer, achieving the highest mean cross-validation score.
5. Inconsistency Detection: The best-performing model (Random Forest) is trained on the entire training set and evaluated on a holdout test set. The test set is used to assess the model's performance in detecting inconsistencies on unseen data. We report several evaluation metrics to provide a comprehensive understanding of the model's effectiveness:
    - Accuracy: The proportion of correct predictions (consistent and inconsistent instances) made by the model.
    - Precision: The proportion of instances predicted as inconsistent that are truly inconsistent.
    - Recall: The proportion of truly inconsistent instances that are correctly identified by the model.
    - F1 Score: The harmonic mean of precision and recall, providing a balanced measure of the model's performance.

The key strength of our solution lies in its ability to leverage machine learning techniques to automatically detect inconsistencies in Wikidata, without relying on manual inspection or rule-based approaches. By training on historical data and incorporating various features, the models can learn patterns and relationships that are indicative of inconsistencies.

However, it is important to note that the performance of the models is heavily dependent on the quality and representativeness of the training data, as well as the effectiveness of the feature engineering process. The introduced random inconsistencies may not fully capture the complexity and diversity of real-world inconsistencies. Additionally, the limited size of the dataset (100,000 instances on our largest run) may impact the generalizability of the models to the entire Wikidata knowledge base.

Evaluation:

To evaluate our solution, we employ cross-validation and other testing techniques. Specifically, we use stratified 5-fold cross-validation to estimate the performance of each classifier (Decision Tree, Random Forest, and SVM) and select the best-performing model. The cross-validation scores for each classifier are reported, along with the mean score.

The Random Forest classifier achieves the highest mean cross-validation score of 0.5074, indicating that it performs better than the other classifiers on average. This suggests that the Random Forest model is able to capture the patterns and relationships in the data more effectively than the Decision Tree and SVM classifiers.

Next, we split the data into train and test sets, with 20% of the data reserved for testing. We train the best-performing classifier (Random Forest) on the training set and evaluate its performance on the holdout test set using the following metrics:

- Accuracy: The proportion of correct predictions (consistent and inconsistent instances) made by the model. An accuracy of 0.522 indicates that the model correctly predicts the consistency status of 52.2% of the instances in the test set.
- Precision: The proportion of instances predicted as inconsistent that are truly inconsistent. A precision of 0.527 means that among the instances predicted as inconsistent by the model, 52.7% are actually inconsistent.

- Recall: A recall of 0.894 implies that the model correctly identifies 89.4% of the truly inconsistent instances in the test set.
- F1 Score: An F1 score of 0.663 indicates a reasonable balance between precision and recall.

Moreover, the evaluation is based on limited sizes ranging from 5,000 to 100,000 instances, which may not fully represent the diversity and complexity of inconsistencies present in the entire Wikidata knowledge base. Furthermore, our analysis is dependent on manually supplying the feature variables, which is not scalable. To assess the generalizability of the solution, it would be beneficial to evaluate the model on larger and more diverse datasets.

Conclusion:

In this project, we proposed a machine learning-based solution to detect inconsistencies in Wikidata, specifically focusing on cases where a person is listed as having a spouse who died before the current date. Our approach involved querying Wikidata, preprocessing the data, feature engineering, and training and evaluating different classifiers.

The Random Forest classifier emerged as the best-performing model, achieving a mean cross-validation score of 0.5074 and an accuracy of 0.522, precision of 0.527, recall of 0.894, and an F1 score of 0.663 on the holdout test set.

One of the key strengths of our solution is its ability to leverage machine learning techniques to automatically detect inconsistencies, without relying on manual inspection or rule-based approaches. By training on historical data and incorporating various features, the models can learn patterns and relationships that are indicative of inconsistencies.

However, our solution also has limitations and areas for improvement. The performance metrics indicate that the models may be prone to false positives (low precision) or false negatives (low recall), depending on the specific use case and desired trade-off between precision and recall.

Overall, this project demonstrates the potential of machine learning techniques in improving data quality and consistency in large-scale knowledge bases like Wikidata. By continuously refining and enhancing the approach.