Pytheas: Pattern-based Table Discovery in CSV Files

Christina Christodoulakis
Dept. of Computer Science
University of Toronto
christina@cs.toronto.edu

Eric B. Munson
Dept. of Computer Science
University of Toronto
ebm@cs.toronto.edu

Moshe Gabel
Dept. of Computer Science
University of Toronto
mgabel@cs.toronto.edu

Angela Demke Brown
Dept. of Computer Science
University of Toronto
demke@cs.toronto.edu

Renée J. Miller
Khoury College of Comp. Sci.
Northeastern University
miller@northeastern.edu

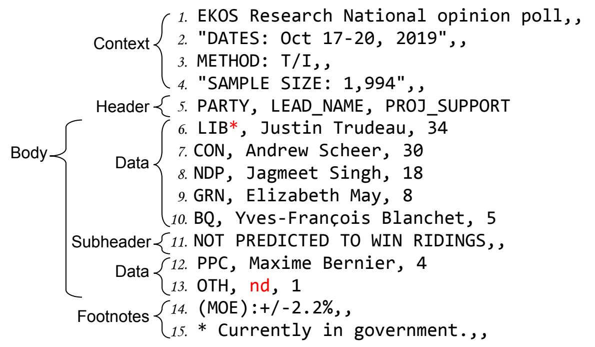
ABSTRACT

CSV is a popular Open Data format widely used in a variety of domains for its simplicity and effectiveness in storing and disseminating data. Unfortunately, data published in this format often does not conform to strict specifications, making automated data extraction from CSV files a painful task. While table discovery from HTML pages or spreadsheets has been studied extensively, extracting tables from CSV files still poses a considerable challenge due to their loosely defined format and limited embedded metadata.

In this work we lay out the challenges of discovering tables in CSV files, and propose Pytheas: a principled method for automatically classifying lines in a CSV file and discovering tables within it based on the intuition that tables maintain a coherency of values in each column. We evaluate our methods over two manually annotated data sets: 2000 CSV files sampled from four Canadian Open Data portals, and 2500 additional files sampled from Canadian, US, UK and Australian portals. Our comparison to state-of-the-art approaches shows that Pytheas is able to successfully discover tables with precision and recall of over 95.9% and 95.7% respectively, while current approaches achieve around 89.6% precision and 81.3% recall. Furthermore, Pytheas's accuracy for correctly classifying all lines per CSV file is 95.6%, versus a maximum of 86.9% for compared approaches. Pytheas generalizes well to new data, with a table discovery F-measure above 95% even when trained on Canadian data and applied to data from different countries. Finally, we introduce a confidence measure for table discovery and demonstrate its value for accurately identifying potential errors.

PVLDB Reference Format:
Christina Christodoulakis, Eric B. Munson, Moshe Gabel, Angela Demke Brown, and Renée J. Miller. Pytheas: Pattern-based Table Discovery in CSV Files . *PVLDB*, 13(11): 2075-2089, 2020.
DOI: https://doi.org/10.14778/3407790.3407810

Proceedings of the VLDB Endowment, Vol. 13, No. 11
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3407790.3407810



```
1.  EKOS Research National opinion poll,,
2.  "DATES: Oct 17-20, 2019",,
3.  METHOD: T/I,,
4.  "SAMPLE SIZE: 1,994",,
5.  PARTY, LEAD_NAME, PROJ_SUPPORT
6.  LIB*, Justin Trudeau, 34
7.  CON, Andrew Scheer, 30
8.  NDP, Jagmeet Singh, 18
9.  GRN, Elizabeth May, 8
10. BQ, Yves-François Blanchet, 5
11. NOT PREDICTED TO WIN RIDINGS,,
12. PPC, Maxime Bernier, 4
13. OTH, nd, 1
14. (MOE):+/-2.2%,,
15. * Currently in government.,,
```

Figure 1: Example CSV file with polling data for the Canadian federal elections of 2019. Line 11 is a subheader, line 6 has a footnote mark referring to line 15, and line 13 contains a missing data mark `nd`.

1. INTRODUCTION

Governments and businesses are embracing the Open Data movement. Open Data has enormous potential to spur economic growth and to enable more efficient, responsive, and effective operation. Within the Open Data world, Comma Separated Value (CSV) files are one of the most important publishing formats [39]. These CSV files form a vast repository of data tables that can be used for query answering [7, 10, 19, 44, 45], text mining [56], knowledge discovery [10, 19, 56], knowledge base construction [14], knowledge augmentation [4, 15, 17, 50, 58], synonym finding [3, 6, 32], and data integration [5, 35, 38, 60, 61] among other tasks [46]. Moreover, CSV files are used extensively across domains, from the environment, to food security, to almost every aspect of government [8].

To unlock the vast potential of Open Data published as CSV files, effective and automated methods for processing these files are needed. Unfortunately, while the CSV format has a standard specification for serializing tables as text (RFC4180) [53], compliance is not consistent. As a result, basic operations like search over Open Data can be difficult and today rely on user-provided metadata.

To illustrate the challenges, consider the simple example CSV file in Figure 1, which is based on a download from `open.canada.ca`. Although the file is well-formed (commas separate values), extracting the data from this file automatically is not trivial. The table body consists of data lines

(lines 6–10 and 12–13) and a single subheader line (line 11) which indicates groupings among the rows in a table. In addition, the file contains a header (line 5), two footnotes (lines 14 and 15), and context lines that describe the contents of the file (lines 1–4). Although not shown in Figure 1, a single CSV file may contain multiple tables, each of which may (or may not) have additional context, header, subheader, and footnote lines associated with the table data. The simplicity and flexibility of the CSV format, which are attractive for Open Data *publishing*, create challenges for Open Data *processing* such as automatic identification of table bodies and their associated headers.

Considerable success has been reported in the research literature for automatic extraction of tabular data from HTML pages [6, 27, 46], and spreadsheets [9, 16, 28, 29, 30]. However, CSV files lack embedded metadata such as formatting and positional information (found in both HTML and spreadsheets) or embedded formulas (found in spreadsheets) that previous methods exploit for the discovery of the table components. Moreover, current state-of-the-art methods focus on inflexible topological rules [9, 28] and machine learning approaches that ignore cell context [29].

Our Contribution: We present Pytheas: a method and a system for discovering tables and classifying lines in CSV files. Pytheas uses a flexible set of rules derived from a study of files from over 100 Open Data portals (Section 2). These rules classify lines based on their cell values, as well as the values of nearby cells. Pytheas is a supervised approach that learns rule weights to best extract tables (Section 3). Our evaluation on 2000 user-annotated CSV files from four Canadian portals (Section 4) shows that Pytheas performs table discovery with a precision and recall of 95.9% and 95.7%, respectively, while current approaches achieve around 89.6% precision and 81.3% recall. We further show that as part of the discovery process, Pytheas performs CSV file annotation with an accuracy of 95.6% compared to a maximum of 86.9% accuracy from prior approaches. An evaluation using over 2500 additional files from four countries shows that Pytheas maintains a high table discovery F-measure (>95.7%), even when trained on data from a single country. Finally, we propose a confidence measure and show that it can be used to reduce labeling effort via active learning.

2. CSV FILES

We briefly introduce the W3C specification for CSV annotations, which we use as a guideline for our work. We follow with a description of our study of a large and diverse set of real CSV files and how it motivated our data model.

2.1 CSV on the Web Specification

As demonstrated in Figure 1, the flexibility of the CSV format can make information extraction difficult. The *W3C CSV on the Web Working Group* defined mechanisms for interpreting a set of CSV files as relational data, including the definition of a vocabulary for describing tables expressed as CSV and describing the relationships between them [12]. They argue that a large fraction of data published on the Web is tabular data in CSV files, and present use cases and detailed publishing requirements aimed at improving usability. Requirements include the ability to add annotations that may be associated with a group of tables, or components of a table at multiple degrees of granularity and support for declaring a missing value token and the reason for missing values (e.g., visual layout, spanning headers, missing or unavailable data, etc.).

Implementing the W3C requirements for CSV requires the ability to identify accurately the main components of tabular data (e.g., header lines, subheader lines, data lines, footnotes and context) in files that contain one or more tables. In this paper, we focus on the identification and annotation of these critical components. We currently support a general model for annotating lines and their cells, but the primitives used in our framework can be extended to support additional W3C specifications.

2.2 Data Model

To develop our table discovery method we performed a large scale study of 111 Open Data portals containing a total of over 600K CSV files. We studied a representative sample of 1798 CSV files covering all portals. Our observations are complementary to the W3C specifications; they both inform the design of Pytheas and confirm prior studies. We use our observations to define a data model for annotation and extraction of data tables and their related components.

A *table* in a CSV file is a contiguous range of lines divided into several contiguous components:

1. **Context** (optional): Tables in CSV files may be preceded by multiple lines with metadata like a title, provenance information, collection methods, or data guides. Based on our observations across portals, we assume such *context* lines typically have values only in the first column. Lines 1–4 in Figure 1 are context lines.
2. **Header** (optional): A *header* is one or more consecutive lines that describe the structure of the table columns. Line 5 in Figure 1 is an example of a header line. We found that while not all tables in CSV files have headers, the majority do. We observed that in some tables the leftmost column(s) functioned as an index while the column's header was empty. In addition, in several portals we observed that the last few columns of a table could be used as comment columns. We further observed that the first table in a file will only lack a header if the table body starts from the first non-empty line. Finally, some multi-line table headers are hierarchical, meaning earlier header lines affect the interpretation of later header lines. We discuss these in detail in Section 3.3.5.
3. **Table body** (required): The table body consists of one or more *data* lines, possibly interspersed with *subheader* lines. Lines 6–13 in Figure 1 make up the table body.
   Data lines are the contents of the table, indicating a row or tuple in the relation that the table represents. Every table must have at least one data line. We observed that cell values do not always conform to the general structure of the column, for example due to null-equivalent values or annotations such as footnote indicators. For example, in Figure 1, lines 6–10 and 12–13 are data lines. The first cell of line 6 contains a footnote character and the second cell of line 13 contains a null-equivalent value.
   Subheader lines describe context for consecutive and adjacent data rows, and can appear interspersed between data lines in the table body. We observed subheaders only in the first column of tables. In Figure 1, line 11 is a subheader.
4. **Footnotes** (optional): As with context, CSV tables are occasionally followed by metadata or clarifications refer-

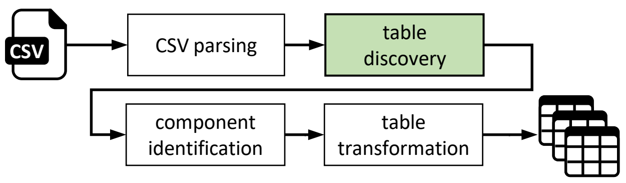


Figure 2: CSV Processing Pipeline: Pytheas focuses on the task of *table discovery* in CSV files.

ring to annotations in the data cells or header cells. Such *footnote* sections can be comprised of multiple lines, and often have non-empty values only in the first column. Consecutive footnotes are often numerically or alphabetically itemized, or use symbols such as '*' and '-'.

Multiple tables: In our study we observed vertically stacked tables in CSV files, but no horizontally stacked tables. Thus, we assume that each line can only belong to one table. This was further confirmed when annotating CSV files in Section 4.1. Moreover, multiple tables in a file were not always separated by blank lines, and did not always have the same headers. Last, we saw that some CSV files do not contain tables. In this case, we label all lines as OTHER.

Table orientation: We assume tables in CSV files are vertically oriented (attributes in columns), as this is by far the most popular format we have observed, consistent with previous research on web tables [15]. Pytheas can be extended to support horizontally oriented tables.

Structure and values: Table structure and content conventions varied by publisher, but could also vary between tables in the same file. We curated a list of common keywords for aggregation (e.g., 'total'), data type specifiers (e.g., 'integer', 'numeric'), range expressions (e.g., 'less than x', 'from x to y'), and popular footnote keywords (e.g., 'note'). We also curated values often used to designate *null-equivalent* values, such as 'nil', 'nd', and 'sans objet'. These values are used to denote missing or redacted data, and typically do not fall into the semantic type expected in the corresponding table column. We use these curated keywords in our fuzzy rule approach (Section 3.2) to help identify lines and cells.

In summary, we identify six distinct classes of non-empty CSV lines: CONTEXT, HEADER, SUBHEADER, DATA, FOOTNOTE and OTHER. Pytheas first discovers tables, and then automatically annotates lines with one of these six classes.

3. TABLE DISCOVERY WITH PYTHEAS

We now introduce Pytheas, our approach for discovering tables in CSV documents such as the example in Figure 1. Table discovery is one of the key steps in the CSV processing pipeline (Figure 2) [13]. Our techniques can also provide building blocks for the next steps in this pipeline, which include identification of table components and table transformations.[1] These steps are out of the scope of this paper, however. We have made Pytheas available at `https://github.com/cchristodoulaki/Pytheas`.

[1]Some approaches extend table discovery to identifying aggregate rows and columns, as well as left headers [1, 9]. We consider these to be part of the table body, and leave identifying them to later stages in the CSV processing pipeline.

3.1 High Level Design

At the heart of Pytheas is a fuzzy rule-based framework, that is based on the knowledge that a data line exists within the context of a table body, and that tables organize information in columns of values that belong to the same semantic concept. Pytheas has an offline phase (i.e., training) and an online table discovery phase (i.e., inference).

In the offline phase, we learn weights for the Pytheas rule-set (Section 3.2). Rules use the spatial context of cells (i.e., the columns and lines to which they belong), and language features of their values to understand the role of each cell.

Figure 3 gives an overview of the online table discovery phase, detailed in Section 3.3. We first parse the CSV file and apply the fuzzy rules to the cells and lines of the file. We use the learned rule weights to compute, for each line, confidence values for it belonging to provisional classes `D` (for data) and `N` (for not data). We then use these line confidences to identify the boundaries of each table body in the file. Given the top boundary of a table and the remaining lines above it, we identify header lines corresponding to the table body, producing the table header. We repeat the table discovery process on remaining lines to discover additional tables, until all lines have been processed.

Pytheas has several parameters, which we define in this section. We discuss how we set them in Section 4.2.2.

3.2 Fuzzy Rules

Fuzzy Rule Based Classification Systems (FRBCS) are widely accepted tools for pattern recognition and classification [25]. They can obtain good accuracy with a relatively small rule set, manage uncertainty and variation in the data effectively, and have been successfully applied to a wide variety of domains such as medical [49], environmental and mining exploration [18], bioinformatics [21] and finance [48] among others. Our inspection of Open Data CSV files revealed that conventions used for table structure have a large number of subtle variations that are difficult to capture exhaustively. In addition, CSV tables frequently contain outlier and null values. In this environment, the flexibility offered by FRBCS in expressing uncertainty, as well as taking into account the context of cells and lines, is important for the development of an effective classifier.

The general form of a fuzzy rule is:

$$\text{Rule } \mathcal{R}_q : \mathcal{A}_q \Rightarrow \mathcal{C}_q, \text{with weight } w_q \in [0,1] \tag{1}$$

where q is the rule index in a rule-set, $\mathcal{A}_q$ is a binary predicate referred to as the rule *antecedent*, and $\mathcal{C}_q$ is the rule *consequent* specifying the predicted class, and w_q is the *weight* given to the prediction. A rule $\mathcal{R}_q$ says that given a condition $\mathcal{A}_q$ is true, w_q is the weight of evidence that the predicted class is $\mathcal{C}_q$.

There are several ways to assign weights to fuzzy rules [26]. Given a labeled data set, the *confidence* of a rule represents the validity of a rule, and is defined as the ratio of instances for which the rule condition was satisfied and the predicted class was correct over the total number of instances for which the condition was satisfied [57]. A simple approach assigns w_q = confidence($\mathcal{R}_q$) [11], however, for highly imbalanced classes a method based on the Penalized Certainty Factor (PCF) has gained increasing acceptance [26].

As our application environment is highly biased to the `D` class, we use the PCF method for assigning weights to rules.

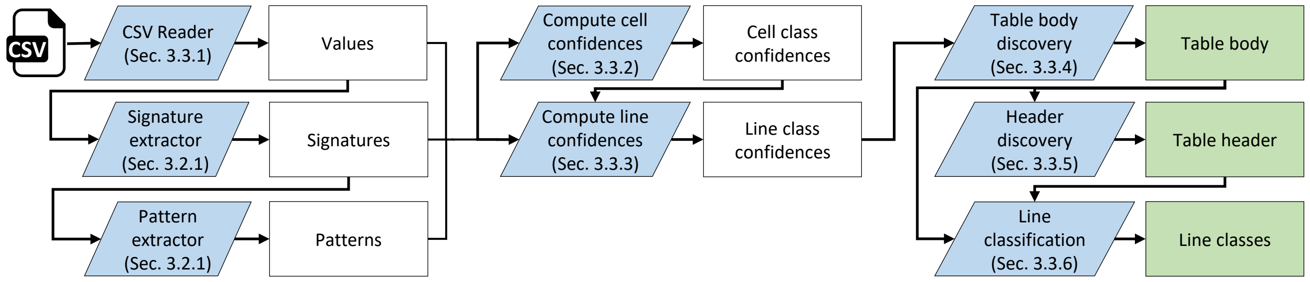

CSV
CSV Reader (Sec. 3.3.1)
Values
Signature extractor (Sec. 3.2.1)
Signatures
Pattern extractor (Sec. 3.2.1)
Patterns
Compute cell confidences (Sec. 3.3.2)
Cell class confidences
Compute line confidences (Sec. 3.3.3)
Line class confidences
Table body discovery (Sec. 3.3.4)
Table body
Header discovery (Sec. 3.3.5)
Table header
Line classification (Sec. 3.3.6)
Line classes

Figure 3: Pytheas's table discovery pipeline. The CSV file is parsed to a 2D array of values. Signature extractors are applied to each value and pattern extractors are applied to signatures. Values, signatures and patterns are used to compute provisional class confidences for cells and then lines. These line confidences are processed to discover table body boundaries, and the top boundary of the table body is used to discover the table header. Finally, all lines of the file are assigned classes.

Table 1: Signatures and their descriptions.

Signature	Description
s_0	Cleaned cell value.
s_1	Run lengths of identical symbol sequences, e.g., s_1(`Justin Trudeau`) = `A6 S1 A7`.
s_2	List of symbols in s_1 (i.e., without lengths).
s_3	Set of symbols in s_2.
s_4	Case of the value as string i.e., `UPPER, lower, Title Case, snake_case, slug-case`, and `camelCase`.
s_5	Number of characters in the value.
s_6	Repetition count of the value in its column context.
s_7	Value subset conforms to range expression.
s_8	Value contains aggregation keywords.
s_9	Value contains footnote keywords.
s_{10}	Value is null equivalent.
s_{11}	Value is a number.

Thus, the weight w_q of rule $\mathcal{R}_q$ is:

$$w_q = \text{confidence}(\mathcal{A}_q \Rightarrow \mathcal{C}_q) - \text{confidence}(\mathcal{A}_q \Rightarrow \overline{\mathcal{C}_q}) \qquad (2)$$

3.2.1 Preliminaries

Before describing rules in our rule-set, we will define concepts that are used to generate rule antecedents.

We discover the encoding and dialect of each CSV file and read it into a matrix of values. Those values are cleaned by removing surrounding quotes and replacing values in a curated *null-equivalent* value dictionary with a new constant that does not appear elsewhere in the file. Let $v_{i,j}$ be the value of a cell. We define the *column context* $T_{i,j}$ of a cell to be a window of κ cells below i, j, and we define the *line context* L_i of a cell i, j to be all cells in the same line.

Cleaned values and their contexts are processed by a **signature extractor** replacing characters with symbols encoding the character type (alphabetic (`A`), digit (`D`), space (`S`), and we treat each remaining punctuation character as its own symbol[2]). For each cell i, j we compute a set of *signatures* $\{s_m(v_{i,j}, T_{i,j})\}$ extracted from the value $v_{i,j}$ and its column context.

Table 1 describes the signatures we use. For example, signature s_7 maps values to predefined regular expressions of ranges that capture values such as "between \$10,000 and \$30,000", "over 75", and "Monday to Friday".

Non-empty signatures of a cell and its context are passed as input to the **pattern extractor**. For each signature type s_m we define two *patterns* that summarize the signatures of a cell and of its context. The pattern P_m is computed from the signature of the cell $s_m(v_{i,j}, T_{i,j})$ and the signatures of its context. In other words, P_m is computed from the signatures of $T_{i,j} \cup \{v_{i,j}\}$ for column rules (Section 3.2.2) and $L_i \cup \{v_{i,j}\}$ for line rules (Section 3.2.3). The pattern P'_m is computed from context signatures only, i.e., from the signatures of $T_{i,j}$ or L_i. Computing patterns from signatures is generally straightforward. For example, the P_1 pattern represents the common prefix of symbols for signatures $s_1(v_{i,j}), ...s_1(v_{i+\kappa,j})$ (Table 1), while P'_1 is the common prefix for $s_1(v_{i+1,j}), ...s_1(v_{i+\kappa,j})$. When symbol counts do not match, they are replaced with `*`.

3.2.2 Column Rules

We specify column-rule antecedents as boolean predicates (conditions) over the signatures of the cell $s_m(v_{i,j}, T_{i,j})$, its column context $T_{i,j}$, and the corresponding patterns P_m, P'_m. Consider the second column in Figure 1. The column context of `'LEAD_NAME'` is `'Justin Trudeau'`, `'Andrew Scheer'`, `'Jagmeet Singh'` in a window of length $\kappa = 3$. The signatures of each value and the corresponding patterns are shown in Table 2. A column rule with the `N` consequent may have an antecedent that translates to *"Signatures from the context [`'Justin Trudeau'`, `'Andrew Scheer'`, `'Jagmeet Singh'`] are summarized to form a pattern that is broken once the signature of `'LEAD_NAME'` is added to the summary"*; a column-rule with the `D` consequent may translate to *"A cell and its context form a strong pattern"*.

Based on our observations described in Section 2.2, we have curated 27 column rules with consequence `D` and 22 column rules with consequence `N`. Table 3 summarizes the list of rules used in Pytheas. Due to space limitations, we briefly list a few example column rules. The full list of column and line rules is available in the GitHub page.

$\mathcal{CR}_1$: Cell and column context are all digits $\Rightarrow$ `D`
$\mathcal{CR}_2$: Column context are all digits but cell isn't $\Rightarrow$ `N`
$\mathcal{CR}_3$: Context prefix has three or more consecutive symbols, but adding cell to the context breaks this pattern $\Rightarrow$ `N`
$\mathcal{CR}_4$: Column context agree on case, cell case disagrees $\Rightarrow$ `N`

[2]Punctuation: !"#\$%&\'() * +, −./ :; <=>?@[']−'{|}

Table 2: Examples of signatures and summarizing patterns for a cell and its column context.

	Signatures				Patterns	
	Cell i, j	**Cells in column-context $T_{i,j}$ of length $\kappa = 3$**			**cell + context**	**context only**
	$v_{i,j}$	$v_{i+1,j}$	$v_{i+2,j}$	$v_{i+3,j}$	P_m	P'_m
	'LEAD_NAME'	'Justin Trudeau'	'Andrew Scheer'	'Jagmeet Singh'		
s_1	A4 _1 A4	A6 S1 A7	A6 S1 A6	A7 S1 A5	A*	A * S1 A*
s_2	A_A	ASA	ASA	ASA	A	ASA
s_3	{A, _}	{A, S}	{A, S}	{A, S}	{A}	{A, S}
s_4	snake_case	Title Case	Title Case	Title Case	$\emptyset$	Title Case

Table 3: Summary of rules for D and N. A full list of rules will be made available in Pytheas github page.

Group	Column D	Column N	Line D	Line N	Total
SYMBOLS: rules that consider structure of a value within a context	22	13	-	4	39
CASE: rules that consider case of a value within a context	2	1	-	2	5
VALUES: rules that consider relationship of values within a context	2	-	1	10	13
LENGTH: rules that consider length of a value within a context	1	8	-	-	9
KEYWORD: rules that use keywords curated from our initial sample	-	-	3	2	5

$\mathcal{CR}_5$: Cell value repeats in the column context $\Rightarrow$ D
$\mathcal{CR}_6$: Cell length $<$ 30% of min cell length in context $\Rightarrow$ N

For example, in Figure 1 column rules $\mathcal{CR}_3$ and $\mathcal{CR}_4$ will both fire for $v_{5,2}$ = 'LEAD_NAME', $\mathcal{CR}_2$ will fire for $v_{5,3}$ = 'PROJ_SUPPORT' and $\mathcal{CR}_1$ will fire on the cell below $v_{6,3}$.

3.2.3 Line Rules

Similarly, we specify line rules as a predicate over the signatures of the cell values in the line and the patterns resulting from those signatures. Line rules depend on the context of the line L_i, as opposed to the column context $T_{i,j}$. As we show in Section 4, corner cases such as header lines and footnotes are important for accurately discovering the boundaries of a table. We give examples of several line rules below. Rules specifically designed to detect header lines are marked with (H), and footnote with (F).

$\mathcal{LR}_1$: First value of a line has aggregation keyword $\Rightarrow$ D
$\mathcal{LR}_2$: Null equivalent value in line $\Rightarrow$ D
$\mathcal{LR}_3$: (H) Line consistently `snake_case` or `camelCase` $\Rightarrow$ N
$\mathcal{LR}_4$: (H) Any value but first is aggregation keyword $\Rightarrow$ N
$\mathcal{LR}_5$: (H) First value starts and ends with parenthesis, and all other cells are empty $\Rightarrow$ N
$\mathcal{LR}_6$: (H) Alphabetic value repeats at a steady interval $\Rightarrow$ N
$\mathcal{LR}_7$: (F) First value of the line matches footnote keyword and third value to end are empty $\Rightarrow$ N

For example, in Figure 1 note that $\mathcal{LR}_2$ will fire on line 13, and $\mathcal{LR}_3$ will fire on line 5. In Figure 6 note that $\mathcal{LR}_6$ will fire on lines $i+1$, and $i+2$, and $\mathcal{LR}_5$ will fire on $i+3$.

3.3 Online Phase: Table Discovery

Once rule weights are trained, Pytheas uses them to discover tables, focusing on accurately discovering table bodies. The Pytheas table discovery process consists of the following steps, each described in a separate subsection.

1. Parse the CSV file into a 2D array of cells and compute the cell signatures for each cell i, j.
2. Compute cell class confidence: apply column rules to signatures of each cell i, j and its column context $T_{i,j}$.
3. Compute line class confidence: apply line rules to each line i and line context L_i and combine these with the cell class confidence for cells in the line.
4. Find the boundaries of a table body (first and last lines).
5. Discover headers for a table.
6. Assign classes to all lines based on tables.

If there are unprocessed lines in the file after a table body and header have been discovered we return to step 4. Occasionally table discovery will find two consecutive tables adjacent to each other, with the second lacking a header. If both have the same number of columns, we merge them.

3.3.1 Parsing and Computing Signatures

Successful processing of CSV files assumes correct parsing (i.e., discovery of file encoding and CSV dialect). We rely on the `chardet` Python library from Mozilla to autodetect encoding [31, 43], and use the standard `csv` Python package for dialect detection. We inspect file content for known structure of HTML, XML and JSON files, automatically identifying and discarding falsely tagged CSV files. We emphasize that the focus of our work is table discovery rather than parsing (illustrated in Figure 2). After parsing, we compute the signatures for each cell as described in Section 3.2.1.

3.3.2 Computing Cell Provisional Class Confidence

Each cell belongs to both the D class, and the N class with a different confidence respectively. We first evaluate the column rules for each cell i, j, as described in Section 3.2.2, and then evaluate the evidence for each possibility. This is depicted by step ① in Figure 4.

Let $u^{(c)}$ be the maximum weight of any column-based rule fired with consequent c, and let $k \in [0, \kappa]$ be the number of non-empty values in the context $T_{i,j}$ for the cell, where κ is the size of the context. We define the *cell class confidence* $z^{(c)}_{i,j}$ for each class $c \in \{\texttt{D}, \texttt{N}\}$ as the maximum of the column

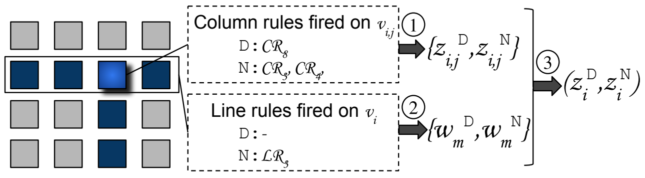

Column rules fired on v_{ij}
D: $\mathcal{CR}_8$
N: $\mathcal{CR}_3, \mathcal{CR}_5$
Line rules fired on v_i
D: -
N: $\mathcal{LR}_5$
$\{z_{i,j}^{D}, z_{i,j}^{N}\}$
$\{w_m^{D}, w_m^{N}\}$
(z_i^{D}, z_i^{N})

Figure 4: Inferring provisional line class confidences $z_i^{(c)}$ from column and line rules applied to cells in line i.

rule weights fired for that class, adjusted with a certainty factor between zero and one representing pattern evidence:

$$z_{i,j}^{(c)} = u^{(c)} \times \left(1 - (1-\gamma)^{2k}\right) \tag{3}$$

Empty cells create uncertainty. Patterns that summarize signatures from a small number of non-empty values will have a lower k and thus lower cell class confidence. The parameter γ depends on the size of the context window, κ. We set γ such that when there are no empty cells, the certainty approaches one.

3.3.3 Computing Line Provisional Class Confidence

Line class inference is a multi-step process (see Figure 4). Once cell class confidences have been collected for all cells in the line ①, we apply line rules and collect rule weights for each line rule fired ②. Cell class confidences and line rule weights are then combined to produce the confidence of a line belonging to each class ③.

We define the *line class confidence* $z_i^{(c)}$ as the fuzzy algebraic sum [37] of all cell class confidences and line rule weights. Let $W = \{w_m^{(c)}\}$ be the set of weights that correspond to line-based rules that have fired over a line i for class $c \in \{\mathtt{D}, \mathtt{N}\}$, then:

$$z_i^{(c)} = 1 - \left(\prod_{j=0}^{J} 1 - z_{i,j}^{(c)}\right)\left(\prod_{m=0}^{M} 1 - w_m^{(c)}\right) \tag{4}$$

where J is the total number of fields in a line and M is the number of line-based rules that fired.

This function has several useful properties. First, the combined class confidence $z_i^{(c)}$ takes values between 0 and 1. Second, the function $1 - \prod_k^K (1 - x_k)$ is an increasing function on x, i.e., the confidence that row i belongs to a class increases when a class confidence of a cell increases. Finally, the function also increases when more cells with class confidences are observed (i.e., evidence of a line belonging to a class increases), meaning the class confidence $z_i^{(c)}$ of line i is greater than or equal to the maximum $z_{i,j}^{(c)}$ since the addition of more evidence for a class increases our class confidence.

3.3.4 Table Body Discovery

The lines i of a file are tentatively assigned a class CS_i by Pytheas, where $CS_i \in$\{D,N\} with a confidence z_i based on the class with the largest confidence: $z_i = \max\{z_i^{\mathtt{D}}, z_i^{\mathtt{N}}\}$.

A table body consists of DATA lines and SUBHEADER lines. The top boundary in a table body can be either a data line or a subheader line. To discover the top boundary, we discover the first data line (FDL), then search for a header section and subheader lines above it. If subheaders exist before the first data line, the top-most subheader becomes the top boundary of the table body. To discover the bottom

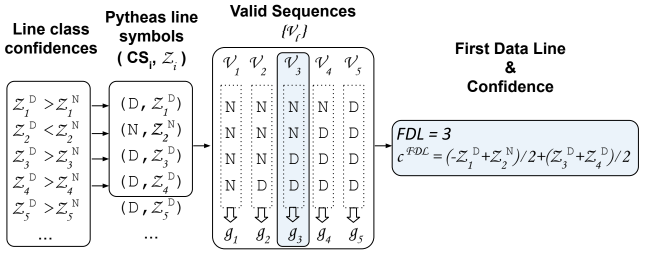

Line class confidences
Pytheas line symbols (CS$_i$, z_i)
Valid Sequences $\{\mathcal{V}_\ell\}$
First Data Line & Confidence
$z_1^D > z_1^N$
$z_2^D < z_2^N$
$z_3^D > z_3^N$
$z_4^D > z_4^N$
$z_5^D > z_5^N$
...
(D, z_1^D)
(N, z_2^N)
(D, z_3^D)
(D, z_4^D)
(D, z_5^D)
...
$\mathcal{V}_1$ $\mathcal{V}_2$ $\mathcal{V}_3$ $\mathcal{V}_4$ $\mathcal{V}_5$
N N N N D
N N N D D
N N D D D
N D D D D
g_1 g_2 g_3 g_4 g_5
FDL = 3
$c^{FDL} = (-z_1^D + z_2^N)/2 + (z_3^D + z_4^D)/2$

Figure 5: Finding the top boundary of the table body. Pytheas tentatively assigns the class with the largest confidence to each line. Line class confidences are used to compute matches g_ℓ between the proposed class sequence and valid sequences in $\{V_\ell\}$. The valid sequence with the best match determines the index of the top boundary of the table body, with a confidence c^{FDL}.

boundary, we search for the first line that is (a) not coherent with existing data lines, and (b) not a subheader followed by coherent data lines.

Inferring Top Boundary: Given confidence pairs $(z_i^{\mathtt{D}}, z_i^{\mathtt{N}})$ for each line i, a straightforward way to discover the top boundary of a table body would be to choose the first line for which confidence $z_i^{\mathtt{D}} \geq z_i^{\mathtt{N}}$, making $CS_i = \mathtt{D}$. However, on occasion, Pytheas may misclassify lines, suggesting a sequence of line labels which is not valid. Figure 5 shows an example where lines 1 and 2 contain a two-line header, but are tentatively classified as D and N respectively. The following two data lines are both tentatively classified as D. However, using line 1 (the first line for which $z_i^{\mathtt{D}} \geq z_i^{\mathtt{N}}$) as the first line of the table body would lead to including the header as part of the body. Pytheas needs to figure out if the misclassification occurred for the first line or the second.

We define a *label sequence* to be a sequence of class labels $Cl_i \in \{\mathtt{D}, \mathtt{N}\}$ of lines $i \in [1, I]$, where i is the index of lines belonging to a table body and the lines preceding it. For the top boundary of a table body, a *valid label sequence* is comprised of zero or more lines of class N followed by zero or more lines of class D. We denote by $Y = \{V_1, V_2, \ldots, V_{I+1}\}$ the set of all valid label sequences of length I, where V_ℓ is a sequence of $I - (\ell - 1)$ lines labeled N followed by $\ell - 1$ lines labeled D. By definition, FDL is the first data line of the table, so there is no point in looking for it deep into the table body. We therefore stop the search for FDL once we provisionally classify θ consecutive lines as D. In the example in Figure 5 the parameter θ is 2, and therefore $I = 4$.

We define the *match* g_ℓ between a candidate valid label sequence V_ℓ and the provisional line classification proposed by Pytheas as:

$$g_\ell = \sum_{i=1}^{I} m_i z_i \tag{5}$$

where $m_i = 1$ if $Cl_i = CS_i$, or -1 otherwise. Pytheas selects as the first data line FDL the Valid Sequence V_ℓ that maximizes the match g_ℓ:

$$\text{FDL} = \underset{\ell \in [1, I+1]}{\arg\max}\{g_\ell\}$$

This maximization criterion improved the performance of Pytheas over the naive criterion of selecting the first D line. In Figure 5, V_3 has the best match so FDL $= 3$.

FDL Confidence: Intuitively, Pytheas can be confident about boundary decisions for which (a) the lines before the boundary confidently agree on their classification (`D` or `N`), and (b) lines after the boundary also confidently agree on their classification. We therefore define *first data line confidence* c^{FDL} as a function of line class confidences in the neighborhood of the first data line:

$$c^{\text{FDL}} = \frac{1}{\alpha+\beta}\left(\alpha \sum_{i=\text{FDL}-1}^{\text{FDL}-\alpha} m_i z_i + \beta \sum_{i=\text{FDL}}^{\text{FDL}+(\beta-1)} m_i z_i\right) \quad (6)$$

where $m_i z_i$ are the confidences of lines i in the best matching valid sequence. Whenever possible $\alpha = \beta$ so that an equal number of lines before and after the predicted first data line contribute to the confidence. We use c^{FDL} as the confidence of the discovered table.

Inferring Bottom Boundary: Once the first data line of the table body (FDL) has been inferred, we find the last line of the table by iterating over lines i starting from the FDL downwards.

For this search, context window size κ becomes the number of confirmed DATA lines from FDL to line i, and context $T_{i,j}$ includes all lines in this window. The bottom boundary is defined as the first line i for which the confidence for `N` outweighs `D`, $z^{\texttt{D}}{}_i < z^{\texttt{N}}{}_i$, unless i is a potential subheader. Note we explicitly skip sub-header lines, allowing them to remain in the table body: A line i is a sub-header if all its cells except the leftmost are empty and either (a) $z^{\texttt{D}}{}_i < z^{\texttt{N}}{}_i$ and $z^{\texttt{D}}{}_{i+1} \geq z^{\texttt{N}}{}_{i+1}$, or (b) the cell value is found in a subsequent cell in the column with an aggregation token. Note that if a line is the bottom boundary, then all lines above it from FDL downwards belong to the table body.

Incremental computation: When searching for the last data line of a table body, we compute patterns P_m (and P'_m) incrementally to avoid quadratic runtime. Since patterns capture the commonality of cell signatures in the context, we can add a new cell to the context incrementally. For example, P'_2 in Table 2 is `ASA`. If we add a cell whose signature s_2 is `AS` then the updated P'_2 will become `AS`. Thus, given the pattern P_m (or P'_m) computed for the current context $T_{i,j}$ (meaning it includes confirmed data lines from FDL to $i-1$), we can incrementally update P_m for the context $T_{i+1,j}$ by adding line i to the context. Pytheas's runtime is therefore linear in the number of lines and columns, which we verify empirically in Section 4.5.

3.3.5 Table Header Discovery

Informed by our observations in Section 2.2, we consider a valid header to be one in which all columns have unique, non-empty header values, with the following exceptions: if the data table spans only two columns, the first column can be empty (without a header); if it spans three or four columns, the first one or two columns can be empty; otherwise, the first and/or last one or two columns can be left empty.

A multi-line header section is a contiguous range of lines that, when merged together, form a valid header. To merge line i with the h lines below it, we prepend each non-empty value to the cells below and to the right. Figure 6 shows how the merge process works. We prepend $v_{i,j}$ to the values in all the cells in the columns j to $\ell-1$ in lines $i+1$ to $i+h$, where ℓ is the column of the next non-empty cell in line i. Multi-line headers may include header metadata at the bottom, defined as lines where all values are enclosed

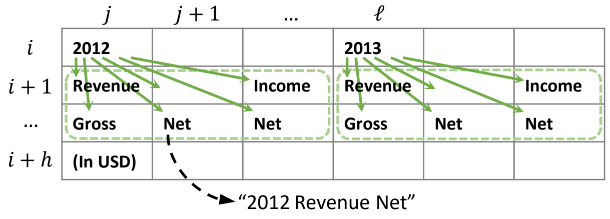


Figure 6: Example of merging hierarchical multi-line headers. The top line is merged with the lines below. The last line is header metadata and is therefore not merged.

in parenthesis or where all but the first columns are empty. Such metadata lines are considered part of the header, but are not merged and checked for validity.

To find the header, we scan lines backwards, starting from the table body towards the beginning of the file or the end of the previous table, and ignoring empty lines, for a maximum of λ non-empty lines. Once a valid header (single line or merged multi-line range) is found, we end the scan. Consecutive header lines adjacent to the top boundary of the table for which only the first cell is not empty are added to the list of subheaders, and the top boundary is adjusted.

If no valid header was found, we relax the uniqueness constraint and search for a single line with non-empty values in all columns. To avoid misclassifying footnote lines as header, we only allow this relaxation for the first table in the file: we observed that files rarely have context lines but no table headers. It is also possible that no header is found at all.

3.3.6 Line Classification

Once table discovery is finished, we assign classes to all lines in the file. If no tables were discovered, then all lines are labeled OTHER. Lines belonging to discovered table components (i.e., header and body) are assigned classes HEADER, DATA and SUBHEADER. Next, non-data lines adjacent to the table bottom boundary are classified as FOOTNOTE and remaining lines are classified as CONTEXT.

4. EXPERIMENTAL EVALUATION

We implemented Pytheas and other state-of-the-art methods using Python [47]. We evaluate Pytheas against alternative methods for table extraction using two large manually-annotated data sets of CSV files.

4.1 Annotated Ground Truth Data Sets

To train and evaluate our models, we construct two ground truth data sets. The **Canadian2K** data set is comprised of 2000 files from CSV resources from Canadian Open Data portals. We use it to evaluate line classification and table discovery performance. The **International** data set contains 2511 files from Canada (491 files across 2 portals, no overlap with Canadian2K), Australia (797 files across 3 portals), the UK (839 files across 5 portals) and the USA (384 files from one portal). We use the International data set to evaluate how well the Pytheas rule-set generalizes.

For every file in the ground truth sample, we detect file encoding and discover the delimiter (if one exists). We manually identify table components and boundary lines of all tables, and label lines accordingly (see Section 2.2). We opted to annotate a larger number of files, rather than annotating complete files, and thus limited our annotation efforts to the first 100 lines per file.

We successfully annotated 1965 valid CSV files from the Canadian2K data set, yielding 2046 tables. The remaining 35 files were corrupt, their encoding could not be discovered, or they were not genuine CSV files (e.g., HTML, JSON, etc.); we discarded these files to ensure that our ground truth data set contained only valid CSV files.

Our annotation efforts confirmed the main observation from our initial study: a large variety of conventions are used in practice, even within the same portal. Ten annotated tables in the Canadian2K data set lacked a header line, but the majority of tables started their data from the second line of the file. Approximately 8% of the tables annotated were accompanied by footnotes. Finally, seven annotated CSV files did not contain tabular data (e.g., cover sheets exported from spreadsheets in CSV).

We also found some notable differences between countries. For example, Canadian data is often bilingual; footnotes often use one column for English and one for French. In contrast, formatting of UK files is less consistent, with multiple tables and footnotes in the same file following different conventions. Finally, unlike the other countries, most files from the USA portal (over 91%) do not include header lines.

4.2 Experimental Setup

We evaluate performance using 10-fold cross validation. Annotated files are randomly shuffled and then partitioned into ten folds to perform ten experiments. Each experiment assigns a different fold as a test set and trains the model on the remaining nine; we average over the performance of all experiments.

Our measures for evaluating performance of our classifiers are *precision* (Equation 7) and *recall* (Equation 8), which are common measures used to describe table extraction performance in the literature [59]. Given instances returned by a method[3] (e.g., lines classified as DATA by the method, or tables extracted), precision estimates the probability that the method is correct (e.g., the lines were truly DATA, or that the tables extracted indeed matched annotated tables):

$$\text{precision} = \frac{\text{number instances correctly evaluated}}{\text{number of instances returned}} \; . \qquad (7)$$

Given the total number of instances in the data set (e.g., the number of DATA lines), recall is the percentage of all cases a method can correctly evaluate:

$$\text{recall} = \frac{\text{number instances correctly evaluated}}{\text{total number of instances}} \; . \qquad (8)$$

Finally, *F-measure* is defined as the harmonic mean of precision P and recall R: F-measure $= \frac{2 \times P \times R}{P+R}$.

4.2.1 Baseline Approaches

We compare Pytheas to the heuristic approach introduced by Mitlöhner et al. [36], which to our knowledge is the only other approach designed specifically for discovering tables in CSV files. Additionally, we compare Pytheas to two state-of-the-art approaches for table extraction from spreadsheets SP_CRF [9] and TIRS [28, 29, 30]. We implement all features that can be extracted from CSV files (unlike, e.g., features that use embedded formatting metadata). We now describe these approaches and our implementations.

CSV Heuristics [36]: Mitlöhner et al. propose a heuristic to approximately isolate headers based on data types of cells as well as the assumption that there are at most two header lines in each file. Lines at the beginning of the file that have zero or one delimiter are considered to be comment lines; multiple tables in a file are found if they have different number of delimiters. We implemented the heuristic-based approach to table detection as described in their paper [36].

SP_CRF [9]: Chen and Cafarella propose automatic extraction of relational data from spreadsheets using linear-chain Conditional Random Fields (CRFs). We extend their work to infer tables from the classified lines. We implement the boolean content-based features described in [9, 45], omitting layout features that are not supported by CSV. We apply a linear-chain Conditional Random Field line classifier using the `CRFsuite` Python library [41]. Each line is represented by a set of features.

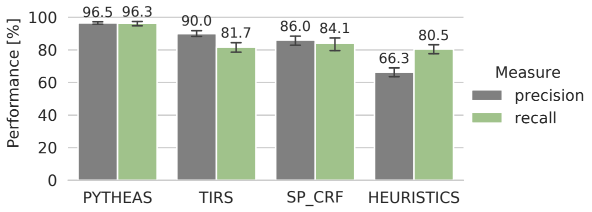
Random Forests and TIRS [28, 29, 30]: Koci et al. focus on the problem of recognizing tables in spreadsheets that may contain multiple tables, stacked vertically or horizontally. They first use a Random Forest (RF) model to classify cells [29], which is followed by TIRS: a topological approach that captures the layout of tables [28, 30] based on cell classes. We implemented those cell-based features described in [29] that can be extracted from CSV files and used `scikit-learn` [42] for the Random Forest implementation. We implemented the TIRS algorithm as described [30], other than leaving out the heuristics and cell features that do not apply to CSV files, and limiting table search to vertically stacked tables to match our data model.

TIRS matches headers and data sections by applying rules on their relative positions in the file. This requires creating and evaluating *composite regions*: regions of discontiguous areas sharing the same label. We found that when the underlying cell classifier creates many discontiguous areas, TIRS's runtime grows exponentially since it evaluates a power set of all possible combinations. In our data, we observed TIRS evaluating at least 2^{35} such combinations in one file, making computation infeasible. We therefore limited composite region evaluation to the largest 15 areas. For header areas, we further require that they overlap by at least one line.

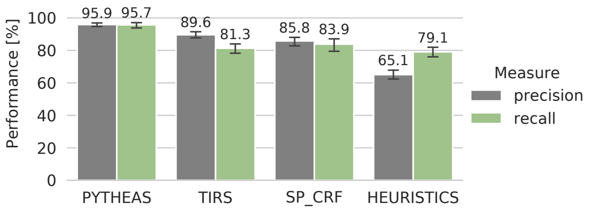
4.2.2 Pytheas Parameter Tuning

Pytheas's parameter values remain fixed throughout all experiments on both Canadian2K and International data. We undersample the DATA class from each annotated file by sampling only two data lines per annotated table. In inference, we limit the input to Pytheas to the 25 leftmost columns in the file. We set the maximum context size κ to 6 by tuning on a 10% subset of the Canadian2K data. Once κ is set, we set γ such that the certainty factor in Equation 3 approaches 1 for $k = \kappa$: γ=0.3. We set the FDL stopping criteria θ to 2. For the FDL confidence we set parameters $\max(\alpha) = \max(\beta) = 4$ to account for possible misclassification of lines in the neighborhood of the table border(Equation 6). We set the maximum non-empty header lines λ to 5 based on our observation of header sizes from the original sample of CSV files (see Section 2.2).

[3]Traditionally, recall and precision are often used for information retrieval and classification tasks. We extend the definition for tasks such as table discovery.



(a) Table Body



(b) Table Body and Header

Figure 7: Table discovery performance and 95% confidence intervals on the Canadian2K data set.

Table 4: Comparison of average line classification performance on the Canadian2K dataset. For each method, we report average and 95% confidence interval for Precision (P) and Recall (R). A '-' indicates the label is unsupported by the method.

	Pytheas		**TIRS**		**SP_CRF**		**Heuristics**	
Class	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**
DATA	**99.93±0.02**	**99.90±0.02**	99.92±0.01	91.94±0.59	99.27±0.11	99.80±0.09	99.03±0.13	99.89±0.01
↳(top only)	**97.37±0.36**	**97.18±0.76**	93.72±0.64	85.01±1.52	86.89±1.40	84.91±2.09	67.94±1.78	80.99±1.55
↳(bottom only)	**98.68±0.33**	**98.48±0.56**	91.88±0.65	83.31±1.30	95.99±0.58	93.71±1.39	80.11±1.60	95.52±1.12
HEADER	**95.13±0.94**	**98.09±0.56**	86.29±3.02	85.76±1.28	88.21±1.98	82.04±1.67	74.15±1.87	82.63±1.86
SUBHEADER	**88.14±3.34**	**88.72±1.98**	-	-	50.00±16.67	0.0 ±0.0	-	-
CONTEXT	**82.94±4.03**	**92.81±1.99**	5.98±1.00	74.35±4.19	67.01±6.28	50.14±4.22	-	-
FOOTNOTE	**91.64±6.51**	**88.68±3.76**	-	-	76.07±9.34	47.29±5.85	-	-
OTHER	**90.0 ±10.0**	60.08±16.30	0.30±0.18	**80.08±13.28**	31.47±14.98	55.00±15.29	90.0 ±10.0	50.08±16.64

4.2.3 *Training*

We cross-validate across files: each of the ten folds in our cross-validation experiment uses 90% of the files in the annotated ground truth as the training set, and the remaining files as the test set. In the training phase of Pytheas we consider only the first table of each annotated file in the training set, taking into account all lines preceding the data section (context and header lines) of the first table and it's first few data lines (effectively under-sampling the DATA class to ensure a balanced training set). We then trained the Pytheas fuzzy rule-set to assign weights to each rule. For TIRS, we use all cell annotations from the first table in each file under-sampling the DATA class with up to three data lines to train a Random Forest model for classifying cells (as in [29]). For SP_CRF, we train on the line-label sequence of all labeled lines from each file (as in [9]).

4.3 Discovery Performance

We evaluate Pytheas and competing state-of-the-art methods on three tasks using the Canadian2k data set: table discovery, line classification, and file annotation. We then evaluate generalizability using the International data set.

4.3.1 *Table Discovery*

We consider two sub-tasks for table discovery: identifying the table body (data and subheader lines), and identifying the combination of body and headers. We use both precision and recall as metrics for correctness in table discovery. We define precision of table discovery as the ratio of tables where the data section was correctly discovered to the number of tables returned by the method. We define recall as the number of tables where the data section was correctly discovered over the number of tables that have been labeled in the ground truth. Since discovered tables might be used for further data analysis, our metrics for correctness are strict: even small errors (such as a single missed line in a large table) mean the entire table was not discovered correctly.

Figure 7 compares table discovery performance, showing averages and 95% confidence intervals across the 10 folds. Pytheas outperforms other methods in both recall and precision, and the difference is statistically significant. As we later show, much of this difference comes from Pytheas's superior identification of table boundaries. Pytheas first finds the table body, followed by the header discovery process, and finally uses these discovered boundaries to assign classes to lines and their cells. In contrast, baseline approaches prioritize line classification, which is then further processed to discover tables and their respective components.

Pytheas's misclassifications generally fall into two categories: very narrow tables whose data cells are similar to non-data cells, and tables shorter than the context κ that are stacked vertically with no separation.

4.3.2 *Line Classification*

To better understand Pytheas' performance, we compare performance of line classification across the four approaches. We measure classification performance for lines for each of the classes in our data model: HEADER, DATA, SUBHEADER, CONTEXT, FOOTNOTE and OTHER. We also separately evaluate the performance of classifying boundary lines (top and bottom data lines of a table), to investigate how table discovery performance depends on boundary identification.

Table 4 lists the performance over the 10-fold cross validation experiments. Pytheas outperforms competing approaches on almost all component classification tasks. Specifically, Pytheas uses table discovery to assign class labels to lines in a file, and then line classification to assign class labels to cells per line. CSV files are highly biased to the

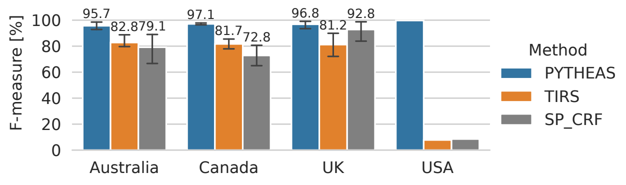


Figure 8: Table discovery performance (Body and Header) averaged over all portals in each country in the International data set. Models are trained on Canadian2K data set.

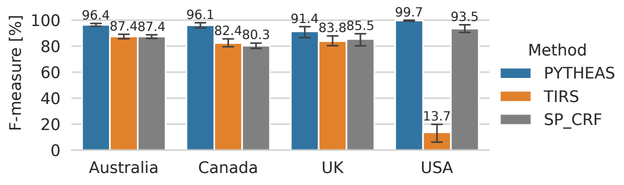


Figure 9: Table discovery performance (Body and Header) with 5-fold cross-validation for different countries in the International data set.

DATA class, making that an easy classification task. Even a small change in classification performance of the DATA class, particularly when it comes to the table body boundaries, propagates to drastic changes in performance for all other classes. This bias towards the DATA class makes it particularly difficult to infer that a given CSV file contains no tables (OTHER class). Pytheas performs well even in this extreme case, while TIRS and SP_CRF achieve very low performance. Interestingly, for this corner case, the heuristic method outperforms the other methods.

For data lines at table boundaries (top and bottom), we see substantial drops in performance for all methods other than Pytheas, comparable to the difference in table discovery performance in Figure 7. Despite very high performance in data line classification across all methods, it is the table body boundary lines that are important. We conclude the gap in performance of table discovery is due to the importance we place on table body boundary classification; the context-aware rules in Pytheas are more successful at detecting table boundaries. Finally, Pytheas sometimes struggles to differentiate between footnotes and context of consecutive tables, leading to lower performance on the CONTEXT and FOOTNOTE classes. Nonetheless, Pytheas outperforms SP_CRF (the only other method that supports both CONTEXT and FOOTNOTE) by a wide margin for these classes.

4.3.3 File Annotation

We consider a CSV file correctly annotated when *all* lines in the file have been correctly classified. We define accuracy of CSV file annotation as the number of files for which all lines were correctly classified divided by the total number of files processed. Pytheas on average annotates CSV files with 95.63% accuracy, compared to 86.43% for SP_CRF, 82.61% for TIRS and 78.74% for the heuristics method.

4.3.4 Ability to Generalize to New Data Sets

We evaluate table discovery performance on the International data set. This data set was not used in Pytheas development: it was annotated after freezing all rules, algorithms, and parameter values.

Figure 8 shows performance of Pytheas and baseline approaches when trained on the Canadian2K data set and tested on the International data set. We observe that the Pytheas rule set is general enough to be trained on data from one country, and applied to data published by other countries. Pytheas maintains high table discovery performance, despite different characteristics of each country's files. This is evident particularly when applied to files from the US portal `catalog.data.gov`, the majority of which do not contain header lines. Figure 9 shows cross-validation performance

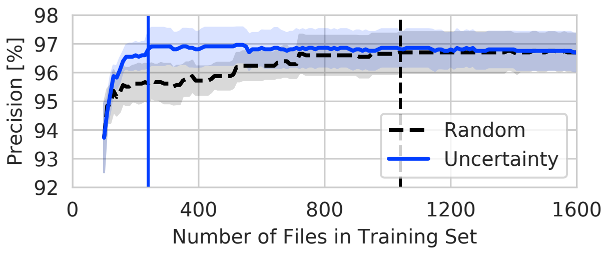


Figure 10: Table discovery precision when growing the training set using active learning (uncertainty sampling) and random sampling. Vertical lines show the training set size at which a method first reaches its final precision (precision when using the entire set of files). Pytheas's confidence measure is more effective at growing the training set, outperforming random sampling and reaching the plateau faster.

when trained and tested on data from the same country. Pytheas continues to achieve competitive table discovery performance compared to state-of-the-art methods designed for spreadsheets. It is interesting to note that TIRS does not perform well in environments where data tables do not have a header, such as files from `catalog.data.gov`, even when trained on the same data.

4.4 Table Discovery Confidence

Unlike TIRS and the heuristic method, Pytheas provides a confidence measure for identified tables (we use the FDL confidence, Eq. 6 in Section 3.3.4). We demonstrate the effectiveness of the confidence measure with an *active learning* experiment. The key idea behind active learning is that a machine learning algorithm can achieve greater accuracy with fewer labeled training instances if it can actively choose the training data from which it learns, as opposed to random selection of training data [51], which reduces the manual effort needed to label new data sets.

Figure 10 shows the results using the Pytheas confidence measure to efficiently build a training set with uncertainty sampling. We shuffled the ground truth data set, randomly selected 400 files to set aside for validation, and used the remaining files as a pool of training data. We initialize our model by training over 10 randomly selected files from the pool of labeled data, and incrementally grow the training set 10 files at a time from the remaining labeled data using either (a) random sampling or (b) *uncertainty sampling*, where files are selected from the pool based on which files the current model is least certain about.

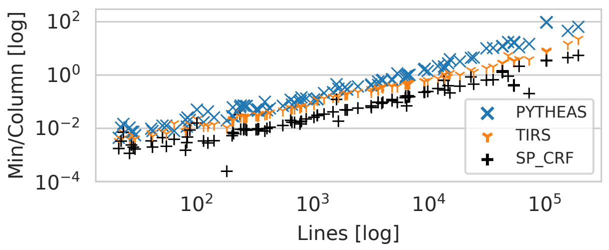


Figure 11: Single core inference time for varied file sizes, normalized to the number of columns in the file.

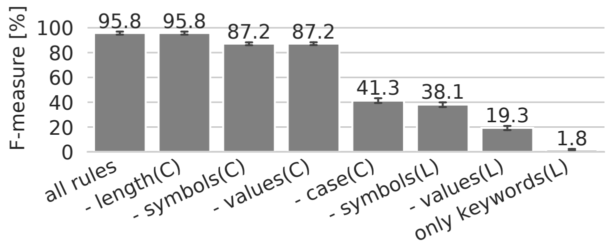


Figure 12: Rule ablation study: F-measure of table Body and Header discovery in Canadian2K data set when sequentially removing groups of rules.

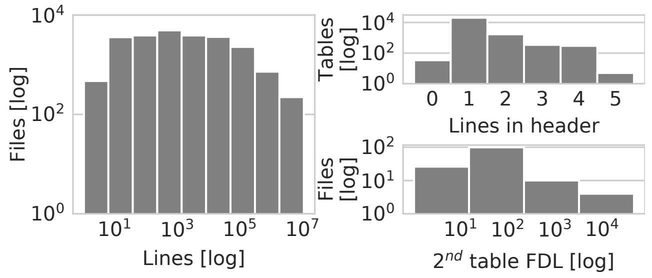


Figure 13: Statistics from one Canadian portal. Left: Distribution of file length for all CSV files in the portal. Top right: header sizes of discovered tables. Bottom right: location of the second table in files with more than one table. We set the confidence threshold for discovering tables to 0.99.

We observe that Pytheas's confidence measure is appropriate for active learning: Pytheas reaches the plateau in precision using only 240 labeled files if a training set is built using uncertainty sampling, while a random sampling approach would require 1040 files. This result represents a significant reduction in the manual effort required to label a data set for training.

While Pytheas performs table discovery (for both body and header) with an average precision of over 96%, some applications that use discovered tables may require input with even higher precision. We find that the majority of discovered tables have a very high confidence, meaning high precision sampling still results in a large proportion of the discovered tables (figure omitted for lack of space). Choosing to disregard tables with confidence < 0.95 ignores only 5% of the data, with the filtered sample's precision at 98.5%.

4.5 Runtime

We measured Pytheas training and inference time on an Intel Xeon E5-4640 CPU running at 2.40 GHz. We process one file per core, and report total runtimes in core-hours. Pytheas used up to 1.8GB of RAM per file processed.

Training: We measured training time over the entire set of 157,420 annotated lines in the Canadian2K file data set. Pytheas takes 9.18 core-hours to train over the entire training set, resulting in 0.21 sec per line. For comparison, TIRS took 0.14 sec per line, while SP_CRF took 0.04 sec per line. Note the limiting factor for training was manual annotation, which was orders of magnitude more expensive.

Inference: Figure 11 shows that inference runtime is linear for Pytheas and compared methods. Pytheas processes files with 0.021 sec per column per line, SP_CRF with 0.005 sec per column per line, and TIRS with 0.011 sec per column per line. This confirms that the incremental computation described in Section 3.3.4 is effective.

4.6 Evaluating Rules

We ran an ablation study on Canadian2K to evaluate the effect of different rules. We grouped similar rules based on what they look at and whether they are line or column rules (see Table 3). Figure 12 plots table discovery performance as we remove successive groups of rules, starting with column rules (C), and followed by line rules (L). Both column and line rules have substantial contributions to performance. Column rules that evaluate structure consistency or case consistency have a large impact on performance since they help differentiate headers from data, and when removed, table discovery dropped by 9 and 49 percentage points respectively. Even in the absence of column rules, line rules can discover well-structured tables by separating header lines from the more common data lines.

We further analyze the coverage and confidence of individual rules. The coverage of a rule is the proportion of cells or lines in the training data on which the rule fires, while confidence tells us how correct a rule is (Section 3.2). Notably, 89% of rules have coverage above 1% of their class proportion, a common threshold for rule coverage [33], suggesting they apply broadly. The few rules with low coverage have high confidence. To evaluate overfitting to the training set, we look at the deviation of rule confidence as calculated over training and test data sets in a 10-fold cross-validation experiment. Except for two rules, the deviation is 5% or lower across all folds. The two rules that show evidence for overfitting do not produce positive weights on this data set, and therefore will not be evaluated during inference.

We conclude that the majority of rules are not specific to a small subset of data, but are broadly applicable, and that the vast majority of rules do not overfit to the training set. When rare overfitting happens, it does not harm performance due to low rule confidence. This is consistent with our generalizability study over International data.

4.7 Table Statistics of a Large Portal

To show the robustness of Pytheas, we applied it to all 23,646 CSV files from the largest Canadian portal that we examined, `open.canada.ca`. Figure 13 (left) shows the distribution of file lengths from this portal. At the time of writing, Pytheas has processed all 22,256 files having up to

181,110 lines per file. We report several interesting statistics from this set.

Of these files, 141 were automatically skipped due to incorrect encoding or invalid (non-CSV) file format. The files used 14 different encodings and 5 different delimiters. Figure 13 summarizes our findings. The majority of files had only one table, but Pytheas discovered up to 226 tables in one file. In total, 25268 tables were extracted from 22,013 files, and 90% of these tables were discovered with $> 99\%$ confidence. Of the high-confidence tables, the widest table discovered had 1787 columns. In addition, 3879 tables had preceding context, with up to 60 context lines for one table, and 1379 tables had footnotes. When a second table existed it was usually found near the beginning of the file (bottom right of Figure 13). The maximum depth at which we found a second table was line 6690. This can be used to optimize processing of very long files by assuming a file will only have one table if we have not seen another in the first 10K lines. We leave such optimizations for future work.

5. RELATED WORK

We categorize related work to those that focus on table discovery, and those that focus on parsing CSV files.

Table discovery: Table discovery is a challenging problem across content types including HTML [6, 27, 46], spreadsheets [9, 16, 30], documents and images [52], plain text [22, 23, 40, 45], and CSV [13, 36]. Mitlöhner et al. [36] analyzed the characteristics of 105K Open Data CSV resources and pointed out that accurate detection of attribute header lines is challenging due to differences in syntactic descriptions of CSV files. They developed heuristics for annotating CSV files and extracting tables from them, but do not present an evaluation of their methods. We find that these heuristics do not apply to many cases in our data set. Koci et al. [29, 30] use Random Forests to classify CSV cells, and a graph representation to capture complex table layouts. Their approach is similar to past methods used to discover relative placement of document images [34]. They present experimental results on spreadsheets from three published corpora, but did not require exact match between the ground truth and the discovered tables.

Pinto et al. [45] first introduced the idea of using Conditional Random Fields (CRFs) for the problem of line classification in plain text documents that contain tables. Adelfio and Samet [1] followed up on this work with a method for extracting the components of tabular data from spreadsheets and HTML tables found on the Web. Chen and Cafarella [9] also used CRFs for discovering tables in spreadsheets. They focus on a common structure of tables in spreadsheets called Data Frames (i.e., tables with a rectangular region of values where only numeric data is located, and regions on the top and/or on the left for attribute headers).

Other works also use the intuition that values in attributes of relational tables exhibit a coherency in structure and semantics [24, 54, 56]. Most recently Chu et al. [10] use coherency to address the problem of transforming HTML lists into multi-column relational tables. They formulate it as an optimization problem, minimizing the pairwise syntactic and semantic distance of potential tuples. This is orthogonal to our work, as their input is a table body and the goal is to align line values to optimize coherency across columns.

Pytheas differs from existing approaches in several ways. First, existing approaches developed for table discovery in HTML pages or spreadsheets leverage the rich feature set that cannot be generated from CSV files, such as spanning cells, formulas, table styling, and text format. For example, TIRS [28, 30] relies on a cell classifier [29] which in turn leverages cell style, formula, and font metadata. These are not present in CSV files, which impacts TIRS's ability to discover tables from CSV files, as shown in Section 4.3. On the other hand, TIRS supports horizontally stacked tables, which we have not observed in CSV files. Second, Pytheas's line classifier takes into account the cell's context: the coherency among the cells in the same line or column. This context may be several lines below a cell, which is particularly important in sparse tables. Third, existing approaches do not focus on the accuracy of identification of the boundary data lines, which are crucial for accurate table discovery. Finally, Pytheas recovers easily from type inconsistencies in table columns (outlier values are common in CSV tables), as its pattern based fuzzy-rules allow for "soft" typing.

CSV parsing and annotation: CSV formatting standards are not followed consistently, leading to much work on properly parsing CSV files. Dohmen et al. [13] turn CSV parsing into a ranking problem with a quality-oriented multi-hypothesis parsing approach. Pytheas is complementary to their work, as they do not place emphasis on table discovery. Ge et al. [20] propose a parser that determines field and record boundaries in arbitrary chunks of CSV data, and detects ill-formed CSV data. Van den Burg et al. [55] developed a consistency measure such that if a file is parsed with the correct parameters the measure indicates higher consistency. Arenas et al. [2] describe a framework for interactively annotating CSV data that takes into account the W3C recommendations for a model for CSV data and the requirements for annotations for CSV data.

6. CONCLUSIONS

While current approaches for automatic table discovery focus on spreadsheet and HTML files, CSV files pose a different challenge due to their lack of rich embedded metadata such as cell formatting, table styling, and formulas. Informed by our observations from a diverse collection of CSV files drawn from over 100 Open Data portals, we designed Pytheas, a fuzzy rule-based approach for table discovery in CSV files. During an offline training phase, Pytheas learns weights for its rule set. These weighted rules are applied in the online inference phase to classify lines, and the line classes with confidences drive the discovery of table boundaries and headers, and confidence of table discovery.

Using two manually annotated data sets with a total of 4500 Open Data CSV files, we show Pytheas's table-focused design outperforms alternative state-of-the-art approaches in table discovery as well as line classification tasks. We also show that Pytheas generalizes well across countries and to new sources of data. Finally, we show that Pytheas's confidence measure can be used to support active learning thereby reducing labeling effort.

7. ACKNOWLEDGMENTS

We thank Periklis Andritsos and Mariano Consens for helpful discussions, and the anonymous reviewers for their detailed and insightful comments. This work was supported in part by an NSERC Discovery grant.

8. REFERENCES

[1] M. D. Adelfio and H. Samet. Schema extraction for tabular data on the web. *PVLDB*, 6(6):421–432, 2013.

[2] M. Arenas, F. Maturana, C. Riveros, and D. Vrgoč. A framework for annotating CSV-like data. *PVLDB*, 9(11):876–887, 2016.

[3] K. Braunschweig, M. Thiele, J. Eberius, and W. Lehner. Column-specific context extraction for web tables. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, SAC '15, page 1072–1077, New York, NY, USA, 2015. Association for Computing Machinery.

[4] K. Braunschweig, M. Thiele, and W. Lehner. From web tables to concepts: A semantic normalization approach. In *International Conference on Conceptual Modeling*, pages 247–260. Springer, 2015.

[5] M. J. Cafarella, A. Halevy, and N. Khoussainova. Data integration for the relational web. *PVLDB*, 2(1):1090–1101, 2009.

[6] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. WebTables: Exploring the power of tables on the web. *PVLDB*, 1(1):538–549, 2008.

[7] M. Canim, C. Cornelio, A. Iyengar, R. Musa, and M. R. Muro. Schemaless queries over document tables with dependencies. *arXiv preprint arXiv:1911.09356*, 2019.

[8] Capgemini Consulting. Creating Value through Open Data: Study on the Impact of Re-use of Public Data Resources. `https://www.europeandataportal.eu/sites/default/files/edp_creating_value_through_open_data_0.pdf`, 2015. Accessed: 2019-09-23.

[9] Z. Chen and M. Cafarella. Automatic web spreadsheet data extraction. In *Proceedings of the 3rd International Workshop on Semantic Search over the Web*, page 1. ACM, 2013.

[10] X. Chu, Y. He, K. Chakrabarti, and K. Ganjam. Tegra: Table extraction by global record alignment. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1713–1728. ACM, 2015.

[11] O. Cordón, M. J. del Jesus, and F. Herrera. A proposal on reasoning methods in fuzzy rule-based classification systems. *International Journal of Approximate Reasoning*, 20(1):21–45, 1999.

[12] CSV on the Web Working Group. CSV on the Web: Use Cases and Requirements. `https://www.w3.org/TR/csvw-ucr/`, 2015. Accessed: 2019-09-14.

[13] T. Döhmen, H. Mühleisen, and P. Boncz. Multi-hypothesis CSV parsing. In *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*, SSDBM '17, pages 16:1–16:12, New York, NY, USA, 2017. ACM.

[14] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 601–610. ACM, 2014.

[15] J. Eberius, K. Braunschweig, M. Hentsch, M. Thiele, A. Ahmadov, and W. Lehner. Building the Dresden web table corpus: A classification approach. In *Big Data Computing (BDC), 2015 IEEE/ACM 2nd International Symposium on*, pages 41–50. IEEE, 2015.

[16] J. Eberius, C. Werner, M. Thiele, K. Braunschweig, L. Dannecker, and W. Lehner. Deexcelerator: A framework for extracting relational data from partially structured documents. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, pages 2477–2480. ACM, 2013.

[17] H. Elmeleegy, J. Madhavan, and A. Halevy. Harvesting relational tables from lists on the web. *PVLDB*, 2(1):1078–1089, 2009.

[18] A. Ford, J. M. Miller, and A. G. Mol. A comparative analysis of weights of evidence, evidential belief functions, and fuzzy logic for mineral potential mapping using incomplete data at the scale of investigation. *Natural Resources Research*, 25(1):19–33, 2016.

[19] W. Gatterbauer, P. Bohunsky, M. Herzog, B. Krüpl, and B. Pollak. Towards domain-independent information extraction from web tables. In *Proceedings of the 16th International Conference on World Wide Web*, pages 71–80. ACM, 2007.

[20] C. Ge, Y. Li, E. Eilebrecht, B. Chandramouli, and D. Kossmann. Speculative distributed CSV data parsing for big data analytics. In *Proceedings of the 2019 International Conference on Management of Data*, SIGMOD '19, pages 883–899, New York, NY, USA, 2019. ACM.

[21] S.-Y. Ho, C.-H. Hsieh, H.-M. Chen, and H.-L. Huang. Interpretable gene expression classifier with an accurate and compact fuzzy rule base for microarray data analysis. *Biosystems*, 85(3):165–176, 2006.

[22] J. Hu, R. Kashi, D. Lopresti, and G. Wilfong. A system for understanding and reformulating tables. In *Proceedings of the Fourth IAPR International Workshop on Document Analysis Systems*, pages 361–372, 2000.

[23] J. Hu, R. S. Kashi, D. P. Lopresti, and G. Wilfong. Medium-independent table detection. In *Document Recognition and Retrieval VII*, volume 3967, pages 291–302. International Society for Optics and Photonics, 1999.

[24] M. Hurst and S. Douglas. Layout and language: Preliminary investigations in recognizing the structure of tables. In *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, volume 2, pages 1043–1047. IEEE, 1997.

[25] H. Ishibuchi, T. Nakashima, and M. Nii. *Classification and Modeling with Linguistic Information Granules: Advanced Approaches to Linguistic Data Mining.* Springer Science & Business Media, 2004.

[26] H. Ishibuchi and T. Yamamoto. Rule weight specification in fuzzy rule-based classification systems. *IEEE Transactions on Fuzzy Systems*, 13(4):428–435, 2005.

[27] Y.-S. Kim and K.-H. Lee. Detecting tables in web documents. *Engineering Applications of Artificial*

Intelligence, 18(6):745–757, 2005.

[28] E. Koci, M. Thiele, W. Lehner, and O. Romero. Table recognition in spreadsheets via a graph representation. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, pages 139–144. IEEE, 2018.

[29] E. Koci, M. Thiele, O. Romero, and W. Lehner. A machine learning approach for layout inference in spreadsheets. In *Proceedings of the International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, IC3K 2016, pages 77–88, Portugal, 2016. SCITEPRESS - Science and Technology Publications, Lda.

[30] E. Koci, M. Thiele, O. Romero, and W. Lehner. Table identification and reconstruction in spreadsheets. In *International Conference on Advanced Information Systems Engineering*, pages 527–541. Springer, 2017.

[31] S. Li and K. Momoi. A composite approach to language/encoding detection. In *Proc. 19th International Unicode Conference*, pages 1–14, 2001.

[32] X. Ling, A. Y. Halevy, F. Wu, and C. Yu. Synthesizing union tables from the web. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[33] B. Liu, Y. Ma, and C.-K. Wong. Classification using association rules: weaknesses and enhancements. In *Data mining for scientific and engineering applications*, pages 591–605. Springer, 2001.

[34] R. C. M. Armon Rahgozar. Graph-based table recognition system, 1996.

[35] R. J. Miller. Open data integration. *PVLDB*, 11(12):2130–2139, 2018.

[36] J. Mitlöhner, S. Neumaier, J. Umbrich, and A. Polleres. Characteristics of open data CSV files. In *Open and Big Data (OBD), International Conference on*, pages 72–79. IEEE, 2016.

[37] M. Mizumoto and K. Tanaka. Fuzzy sets and their operations. *Information and Control*, 48(1):30–48, 1981.

[38] F. Nargesian, E. Zhu, K. Q. Pu, and R. J. Miller. Table union search on open data. *PVLDB*, 11(7):813–825, 2018.

[39] S. Neumaier, A. Polleres, S. Steyskal, and J. Umbrich. Data integration for open data on the web. In *Reasoning Web International Summer School*, pages 1–28. Springer, 2017.

[40] H. T. Ng, C. Y. Lim, and J. L. T. Koo. Learning to recognize tables in free text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 443–450. Association for Computational Linguistics, 1999.

[41] N. Okazaki. CRFsuite: A fast implementation of conditional random fields (CRFs), 2007.

[42] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[43] M. Pilgrim, D. Blanchard, and I. Cordasco. Chardet.

[44] R. Pimplikar and S. Sarawagi. Answering table queries on the web using column keywords. *PVLDB*, 5(10):908–919, 2012.

[45] D. Pinto, A. McCallum, X. Wei, and W. B. Croft. Table extraction using conditional random fields. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information retrieval*, pages 235–242. ACM, 2003.

[46] J. C. Roldán, P. Jiménez, and R. Corchuelo. On extracting data from tables that are encoded using HTML. *Knowledge-Based Systems*, 2019.

[47] G. Rossum. Python library reference. Technical report, Amsterdam, The Netherlands, The Netherlands, 1995.

[48] J. A. Sanz, D. Bernardo, F. Herrera, H. Bustince, and H. Hagras. A compact evolutionary interval-valued fuzzy rule-based classification system for the modeling and prediction of real-world financial applications with imbalanced data. *IEEE Transactions on Fuzzy Systems*, 23(4):973–990, 2014.

[49] J. A. Sanz Delgado, M. Galar Idoate, A. Jurío Munárriz, A. Brugos Larumbe, M. Pagola Barrio, and H. Bustince Sola. Medical diagnosis of cardiovascular diseases using an interval-valued fuzzy rule-based classification system. *Applied Soft Computing 20 (2014) 103–111*, 2013.

[50] Y. A. Sekhavat, F. Di Paolo, D. Barbosa, and P. Merialdo. Knowledge base augmentation using tabular data. In *Workshop on Linked Data on the Web*, 2014.

[51] B. Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[52] F. Shafait and R. Smith. Table detection in heterogeneous documents. In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, pages 65–72. ACM, 2010.

[53] Y. Shafranovich. Common format and mime type for comma-separated values (CSV) files. 2005.

[54] K. M. Tubbs and D. W. Embley. Recognizing records from the extracted cells of microfilm tables. In *Proceedings of the 2002 ACM symposium on Document Engineering*, pages 149–156. ACM, 2002.

[55] G. J. van den Burg, A. Nazabal, and C. Sutton. Wrangling messy CSV files by detecting row and type patterns. *arXiv preprint arXiv:1811.11242*, 2018.

[56] Y. Wang and J. Hu. Detecting tables in HTML documents. In *International Workshop on Document Analysis Systems*, pages 249–260. Springer, 2002.

[57] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal. *Data Mining*, chapter 3. Output: knowledge representation. Morgan Kaufmann, 2017.

[58] M. Yakout, K. Ganjam, K. Chakrabarti, and S. Chaudhuri. Infogather: Entity augmentation and attribute discovery by holistic matching with web tables. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 97–108. ACM, 2012.

[59] R. Zanibbi, D. Blostein, and R. Cordy. A survey of table recognition: Models, observations, transformations, and inferences. *Int. J. Doc. Anal. Recognit.*, 7(1):1–16, Mar. 2004.

[60] M. Zhang and K. Chakrabarti. Infogather+: Semantic matching and annotation of numeric and time-varying attributes in web tables. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 145–156. ACM, 2013.

[61] E. Zhu, F. Nargesian, K. Q. Pu, and R. J. Miller. LSH ensemble: Internet-scale domain search. *PVLDB*, 9(12):1185–1196, 2016.