# Active Learning for Spreadsheet Cell Classification

Julius Gonsior, Josephine Rehak, Maik Thiele, Elvis Koci, Michael Günther and Wolfgang Lehner
Technische Universität Dresden
firstname.lastname@tu-dresden.de

## ABSTRACT

Spreadsheets are mainly the most successful content generation tools, used in almost every enterprise to create a plethora of semi-structured data. However, this information is often intermingled with various formatting, layout, and textual metadata, making it hard to identify and extract the actual tabularly structured payload. For this reason, automated information extraction from spreadsheets is a challenging task. Previous papers proposed cell classification as a first step of the table extraction process, which, however, requires a substantial amount of labeled training data, that is expensive to obtain. Therefore, in this paper we investigate a semi-supervised approach called Active Learning (AL), that can be used to train classification models by selecting only the most informative examples from an unlabeled dataset. In detail, we implement an AL cycle for spreadsheet cell classification by investigating different selection strategies and stopping criteria. We compare the performance of various AL strategies and derive guidelines for semi-supervised cell classification. Our experiments show, that by implementing AL for cell classification, we are able to reduce the amount of training data by 90% without any accuracy losses compared to a passive classifier.

## KEYWORDS

Information Extraction, Active Learning, Semi-supervised, Machine Learning, Classification, Spreadsheets

## 1 INTRODUCTION

Spreadsheets are powerful content generation tools, assisting novices and professionals alike. They contain data that are roughly relational, but accompanied by various formatting, layout, and textual metadata. Thus, the generated content is primarily designed for human consumption and carries a lot of implicit information. Due to these reasons, automatic table extraction and recognition [2, 5, 17] for spreadsheets is a very difficult task. The most crucial step for all table recognition approaches is cell classification, determining for each non-empty cell its role (data, header, metadata etc.) within the spreadsheet. However, the training of classification models relies on human-labeled training data, which involves extensive human effort. In this paper, we therefore propose a semi-supervised approach for cell classification. In detail, we propose an active learning (AL) cycle that is able to determine the optimal amount of training data, needed to train a cell classification model.

Figure 1 sketches the AL cycle and its main steps. The AL process starts with two sets of data: the unlabeled sample set $U$ and the already labeled dataset $L$, with $|L| << |U|$. Initially, the learner is trained on the small labeled dataset $L$. The main task of the AL cycle is to iteratively increase the set of labeled data $L$ by identifying the most promising cells in $U$. The cycle should stop as soon as the trained classifier reaches the best accuracy. The AL cycle is performed by two main actors: a learner and an oracle. In our case, a learner is a cell classifier which is continuously retrained on the newly labeled data $L_q$. The oracle maintains the label-providing entity, i.e. a human annotator providing the gold standard. Based on the utilized *query strategy* (see Section 3.1) an unlabeled sample $U_q$ is chosen from the unlabeled pool $U$. The purpose of this task is to identify such cells that contribute most to the classifier to be trained. All other cells remain unlabeled, potentially saving a lot of human effort. The cells proposed by the query strategy $U_q$ are then given to the oracle resulting in $L_q$. The newly labeled data $L_q$ is added to $L$ and the process starts again by retraining the classifier on the extended dataset. The AL cycle proceeds until a *stopping criteria* (see Section 3.2) is met or until $U$ is out of queries.

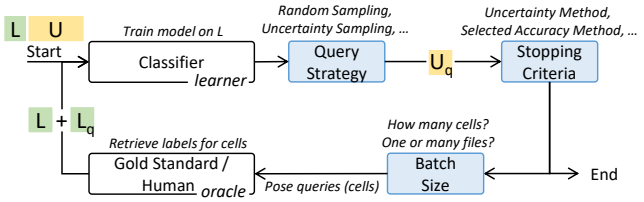


**Figure 1: Active Learning Cycle for Cell Classification**

Our results show, that the classification models trained on the data determined by the AL cycle outperform the passive classifier and reduce the amount of training data by 90%.

**Outline.** The remainder of this paper is organized as follows: In Section 2, we shortly review the task of cell classification for spreadsheets. The individual parts of the AL cycle for cell classification are introduced in Section 3. In detail, we discuss different *query strategies*, *stopping criteria* and *batch sizes*, evaluated in Section 4. We additionally evaluate varying start set sizes and give a recommendation on how to implement the AL cycle for cell classification. Finally, we present related work in Section 5 and conclude in Section 6.

## 2 CELL CLASSIFICATION FOR SPREADSHEETS

The objective of capturing the tabular data embedded in spreadsheets can be treated as a classification problem where the specific structures of a table have to be identified. Given a spreadsheet the goal is to classify each non-empty cell with a given set of labels. The set of labels we use in this work is described in Section 2.1. Section 2.2 briefly presents the cell features used to train a classifier.

### 2.1 Cell Labels

The research literature [2, 5, 33] basically defines seven roles for non-empty cells: *Data*, *Header*, *Derived*, *GroupHeader*, *Title*, *Note*, and *Other*. However, for the specific task of table identification not

all of these roles are equally important and most approaches just rely on the main table building blocks: header, data, and metadata cells. Header cells give names to columns, describing the values below them. They can be nested occupying several consecutive rows. Data cells follow the structure defined by header cells and contain the table's payload. Metadata cells provide additional information about the sheet as a whole, or about a subset of its values. Some typical examples are footnotes and table titles.

## 2.2 Cell Features

The label of a cell is determined by a classifier, which makes use of cell features such as formatting, content types, formula references, and additional context from nearby cells [2, 5]. In total, we consider 159 available features to train our cell classifiers. A detailed explanation of all cell features can be found in [19].

## 2.3 Training Dataset

For our experiments we used the DECO dataset [18], an existing corpus of labeled spreadsheets, that can be used to train classification models and additionally, in context of our work, to simulate an oracle (for more information see the Section 3) in the AL cycle. DECO contains 1,165 labeled spreadsheets which have been derived from the Enron corpus [13]. The time to annotate a sheet within a spreadsheet has been logged during the creation of DECO. In average the annotation takes 4.3 minutes per spreadsheet with a max value of 284.4 minutes. This shows the complexity of the cumbersome manual labeling task. It is clear, that a reduction of to-be-labeled spreadsheets will drastically improve the cell classification approach and thus the overall table extraction process.

# 3 ACTIVE LEARNING FOR SPREADSHEET CELL CLASSIFICATION

To implement an AL cycle (shown in Figure 1) for a given supervised machine learning task, such as cell classification, one has to decide for a *query strategy*, a *stopping criteria* and a certain *batch size*. The *query strategy* selects the samples which should be labeled next by the oracle. In Section 3.1, we give an overview of some popular query strategies which are later considered in our evaluation. To achieve our goal of reducing the human effort in labeling spreadsheet cells, we have to find the optimal point for stopping the AL cycle. Therefore, different stopping criteria are discussed in Section 3.2. Another issue that impacts the labeling effort is the batch size, i.e. the number of samples or cells labeled within each iteration of the AL cycle, discussed in Section 3.3.

## 3.1 Query Strategies

In this section, we shortly introduce the different strategies for choosing the most informative queries. Each strategy approximates the contained informativeness of unlabeled data for a potential classifier.

*Random Sampling. Random sampling* is a common AL query strategy and found application in [3, 7, 27]. Unlike the other methods, random sampling chooses queries at random and fully independently of their informativeness. However, even with this strategy a rise in prediction accuracy is possible, since the amount of training data is steadily increased. We use random sampling as a baseline to compare the other strategies, too.

*Uncertainty Sampling. Uncertainty sampling* chooses queries which are the most uncertain to predict. Hence, learning these queries should result in more certain predictions of the classifier. We compare three uncertainty metrics: least confident, margin sampling and entropy [28]. Least confidence [20] tries to capture the probability, that the classifier is mislabeling the data using the posterior probability $P$ where $\hat{y}$ is the most likely prediction:

$$U_{q,LC} = \underset{x}{\operatorname{argmax}}\, 1 - P(\hat{y}|x), x \in U \tag{1}$$

Information about other classes next to the most probable one is not taken into account by this strategy. Margin sampling [25] in contrast uses the posteriors for the first and second most probable classes and samples the instances with the smallest margin:

$$U_{q,M} = \underset{x}{\operatorname{argmin}}\, P(\hat{y}_1|x) - P(\hat{y}_2|x) \tag{2}$$

Entropy uncertainty [30] uses all possible classes and captures the entropy of a given distribution:

$$U_{q,E} = \underset{x}{\operatorname{argmax}} - \sum_i P(y_i|x) \log P(y_i|x) \tag{3}$$

*Query-by-Committee.* In contrast to the other strategies the Query-by-Committee [29] approach maintains multiple models in parallel, called committee members. They are all being trained on the same current labeled set $L$ and then vote on the query candidates. The vote is conducted by letting all committee members predict all unlabeled samples and measuring the controversial score for each sample. The most controversial query is considered the most informative. For the measurement of disagreement various methods exist. In this paper, we choose *vote entropy* as proposed by [9]:

$$U_{q,VE} = \underset{x}{\operatorname{argmax}} - \sum_i \frac{V(y_i)}{C} \log \frac{V(y_i)}{C} \tag{4}$$

Whereby $C$ denotes the committee size, $y_i$ ranges over all possible labels and $V(y_i)$ is the number of times a label was predicted by a classifier. This scoring method scores dissenting votes as the highest and concurring votes as the lowest. Our applied committee consisted of three Random Forest Classifiers with different random initializations, one Naïve Bayes, and one Support Vector Machine.

## 3.2 Stopping Criteria

So far, the AL cycle would stop only, if the set of unlabeled data $U$ is out of cells. Obviously this leads to no reduction in labeling effort. Therefore, we introduce a *stopping criteria*, that is able to detect whether proceeding the AL cycle is not resulting in any accuracy gain. Additionally, it can be shown, that by reducing the amount of training data, we prevent overfitting and the test accuracy is increased (Section 4.6). In this paper, we investigate three different stopping criteria (SC) from [34].

*Maximum Uncertainty. Maximum Uncertainty* uses the same measurement as the *Uncertainty Sampling Least Confident* strategy in Section 3.1. The basic idea is, that by passing multiple learning cycles the classifier's uncertainty should decrease as the classifier has more knowledge over its data. Therefore, the stopping criteria uses the uncertainty of the most uncertain sample in $U$ as measure. If the uncertainty value drops below a user-defined threshold, the stopping criteria will end the AL cycle:

$$SC_{MU} = \begin{cases} 1 & , U_q \in U \text{ and } UM(U_q) \leq \theta_{MU} \\ 0 & , \text{otherwise} \end{cases} \tag{5}$$

$UM(x^*)$ is the method retrieving the uncertainty of a sample $U_q$ and $\theta_{MU}$ the predefined threshold. For batches we used the

smallest uncertainty of the batch, other common strategies are the mean or the median.

*Selected Accuracy Method.* This method focuses on the classification accuracy of the chosen queries $U_q$. It assumes, that the classifier always chooses the most difficult queries for labeling. Hence the model's predictions on the most uncertain queries should reflect the classifier's certainty. The AL process stops, if the classifier successfully predicts the labels of those most difficult queries.

$$SC_{SA} = \begin{cases} 1 & , acc_{|U_q|}(C) \geq \theta_{SA} \\ 0 & , \text{otherwise} \end{cases} \tag{6}$$

$acc_m(C)$ determines with the help of the oracle the accuracy of the classifier's predictions on the queries, $m$ denotes the iteration of the AL cycle of the query. $\theta_{SA}$ is the threshold. For batches the mean accuracy for each cell is used.

*Standard Deviation-based Stopping.* In contrast to other approaches, Standard Deviation-based Stopping [6] assesses multiple cycles for trend detection. Two criteria have to be met for stopping: first, the standard deviation of the accuracies for the last five $U_q$ has to be lower than a predefined threshold $\theta_1$ and secondly, the accuracy of the current $U_q$ has to be larger than the threshold $\theta_2$. The first criteria identifies plateaus in the query accuracy and the second one prevents local optima.

$$SC_{\sigma} = \begin{cases} 1 & , \sigma(acc_{n-5\ldots n}) \leq \theta_1 \wedge acc_n \geq \theta_2, \text{when } n \geq 5 \\ 0 & , \text{otherwise} \end{cases} \tag{7}$$

The current cycle under inspection is denoted with $n$.
Since the stopping criteria described above all return Boolean values, they can be used in combination in form of logical expressions.

### 3.3 Batch Sizes

It is common practice in machine learning to train a model on batches of samples instead of single data points. In the later evaluation (Section 4.4) we compare simple top-$k$ batches and so-called *spreadsheet batches*: A top-$k$ batch consists of the top-$k$ cells being selected by the query strategy (Section 3.1). However, this strategy has the disadvantage, that the selected cells potentially belong to a large number of spreadsheets which leads to higher efforts for the oracle, i.e. the human annotator. As reported in Section 2.3 it takes several minutes to label a spreadsheet, while most of the time is needed to understand the spreadsheet content and structure. For this reason, we propose to provide just one spreadsheet to the human annotator within an AL cycle, i.e. a batch consists of all cells from this single spreadsheet. To select the best spreadsheet within an AL cycle we have to adapt our defined query strategy metrics $QS$ (see Section 3.1) for spreadsheet batches, where $U_i$ denotes the set of unlabeled cells of spreadsheet $i$ as follows and $SB$ denotes the spreadsheet to be used as $U_q$:

$$SB = \underset{i}{\text{argmax}} \frac{1}{|U_i|} \sum_{x \in U_i} QS(x) \tag{8}$$

## 4 EVALUATION

To evaluate our semi-supervised approach for spreadsheet cell classification, we performed a set of experiments on DECO dataset (Section 2.3). To use this dataset for cell classification three changes have been made: 1) As described already in Section 2.1 the seven layout roles have to be merged into the three broader categories *data*, *header*, *metadata*. 2) Due to the very high amount of *data* cells (94%) in DECO, the data class would get an increased prior probability. Therefore, we decided to limit the number of data cells to 100 per spreadsheet, chosen randomly. 3) Cells with the label *other* were not included in our experiments, since they are too ambiguous and not helpful for table identification. The original dataset contains 98.68% *data* cells, 1.18% *Header* cells and 0.14% *metadata* cells, with a total of 1,393,398 cells. After performing the aforementioned steps, the distribution is 83.27% *data* cells, 14.99% *header* cells and 1.74% *metadata* cells.
We implemented the AL cycle sketched in Figure 1 in Python in the following way: the *learner* consists of several classification models such as Decision Trees, Random Forest, Naïve Bayes and SVMs, using their Scikit-learn [22] implementation. To perform AL on the DECO dataset we split it into 80% used for $L$ and $U$ as well as 20% for testing. Within the experiments, we usually set $|L|$ to 1% and $|U|$ to 99% (of the overall 80%). Since our dataset is already completely annotated, our implemented oracle can automatically provide the requested label, i.e. for a given $U_q$ provide $L_q$. While this setup is perfect to test the different configurations of the AL cycle it is important to note, that in a real-word setting the oracle will always be a human annotator.
In Section 4.1, we first propose a metric to measure the performance of an AL cycle in terms of accuracy improvement and sample size reduction. Before looking at the AL strategies, we report in Section 4.2 on the accuracy of passive learners, i.e. classifiers using the whole DECO dataset for training. In Section 4.3, we study the impact of different start set sizes on the learning curves. Varying batch sizes, sampling strategies and stopping criteria are investigated in Section 4.4, Section 4.5 and Section 4.6.

### 4.1 Slope Calculation

There does not exist a standardized numeric measure to report on AL performances. A common method in AL research is to compare smoothed accuracy graphs. However, to compare different settings in a faster way, many papers also rate the classifier performance by looking at the start and end point of the learning curve. To take into account also the reduction of the human effort in labeling data, we additionally divide the gained accuracy by the numbers of queries, that have been used to achieve this result. The measurement is denoted as slope.

$$slope = \frac{acc_{end} - acc_{start}}{|queries|} \tag{9}$$

The $acc_{start}$ is the classifier's accuracy before any AL cycle passes. The $acc_{end}$ is determined by the stopping criteria (see Section 3.2) or in the worst case corresponds to the point when the unlabeled sample set $U$ is out of data.

### 4.2 Passive Learning

In a first experiment, we compared the performance of different classifiers trained on the whole dataset $L$, without applying the AL cycle. In detail, we investigated Decision Trees (DT), Random Forest (RF), Naïve Bayes (NB), Support Vector Machines (SVM), and Multi-Layer-Perceptron (MLP) using a standard random search over the hyperparameter and five-fold cross-validation. Table 1 lists the achieved classification results. The data has been downsampled and the data points of the underrepresented classes have been used as weights for the accuracies for the header and metadata cells, as the F1-Sores without would be significantly lower compared to the F1-Scores of the data class. Even though

| | Decision Tree | Naïve Bayes | Random Forest | SVM | MLP |
|---|---|---|---|---|---|
| Data | 0.95 | 0.91 | 0.95 | 0.95 | 0.95 |
| Header | 0.75 | 0.69 | 0.76 | 0.78 | 0.78 |
| Metadata | 0.23 | 0.35 | 0.46 | 0.47 | 0.52 |
| Weighted Avg. F1-Score | 0.90 | 0.87 | 0.91 | 0.92 | 0.92 |
| Avg. training time (sec) | 0.15 | 0.09 | 0.33 | 112.52 | 490.91 |

**Table 1: F1-Scores of the Passive Classifiers Problem**

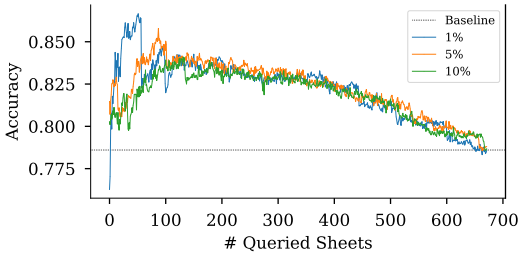


**Figure 2: Comparison of different start sizes and the passive classifier as baseline**

MLP and SVM achieved the best average F1-score we used the close second best option RF for the remaining AL experiments, since its training time is significantly lower. In the following experiments the shown baseline is the accuracy of a passive classifier trained on the whole dataset $L$. Table 1 lists the achieved classification results.

## 4.3 Start Set Size

We compare three different start set sizes $L$: 1%, 5% and 10%, the batch size is 1 spreadsheet at a time, and the query strategy *entropy based uncertainty*. Figure 2 shows, that a smaller start set size leads to higher increase in accuracy at the beginning of the AL cycle. Nevertheless, after around 100 queried sheets for all different start sizes the graphs have almost the same shape. In terms of reducing human effort, we can therefore state, that a small start size is more beneficial and leads to the same or even higher accuracies.

## 4.4 Batch Size

Instead of querying the oracle for just one new labeled data point during a single AL cycle, the classifier is being refitted on a whole batch of new labeled cell sheets. Different batch sizes, respectively 50, 150, 250 and all cells of a *single spreadsheet* are selected for comparison. The start set size for this experiment is 1% and entropy based uncertainty query strategy is used (Section 3.1). Figure 3 shows, that the best results can be achieved with a batch size of 50. However, the difference between cell-based batches and the batches consisting of cells, that belong to a single spreadsheet is very small. Therefore, we advise to use a batch size of one *single spreadsheet* per AL cycle in order to reduce the effort for the human annotator.

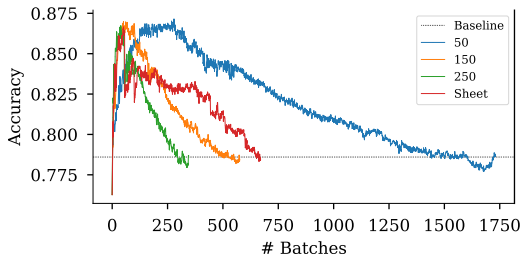


**Figure 3: Comparison of different batch sizes and the passive classifier as baseline**

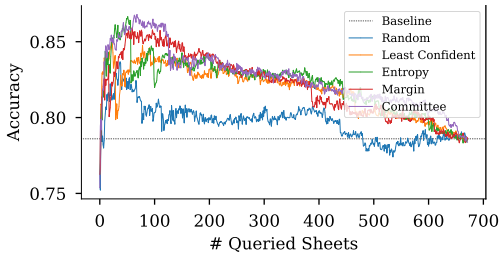


**Figure 4: Accuracies for different sampling strategies and passive classifier as baseline**

## 4.5 Sampling Strategies

Figure 4 shows the classification accuracies for the different sampling strategies, all with the same start size of 1% and spreadsheet-wise batches. One can see a clear difference between the random strategy and the other strategies: For the *random strategy* the accuracy goes down after a few queried sheets. *Entropy based uncertainty sampling*, *uncertainty margin sampling* and *committee sampling* show the highest increase while the latter achieved the best results and is therefore the recommended strategy.

## 4.6 Stopping Criteria

For evaluating the stopping criteria, we use a start size of 1% and committee sampling. Figure 5 shows the values of the three stopping metrics proposed in Section 3.2. The markers indicate the stopping points resulting from these stopping metrics. The vertical dashed line denotes the optimal stopping point derived from the peak of the learning curve for the test data. It should be noted again, that the learning curve is just available in our experimental setting. In practice the stopping point has to be determined by the stopping criteria, that just rely on the output of the classifier.

For sheet-based batches, the *maximum uncertainty* stopping values are quite consistent, only for the last queries a drop is noticeable. The reason is probably an overall consistent average uncertainty over all unlabeled sheets. The *selected accuracy* metric fluctuates quite a lot and depends heavily on the sheet, but an upward trend is nevertheless noticeable the more labeled training data becomes available. The *standard deviation* of the last five classification accuracies of the queried spreadsheets is quite stable, having some valleys. This stopping criterion provides a robust categorization of the accuracies, but cannot be used to distinguish between plateaus and valleys. Even though in our case the minimum of the standard deviation is almost exactly at

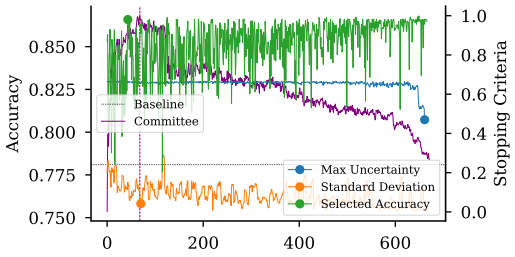


**Figure 5: Comparison of stopping strategies *Max Uncertainty*, *Standard Deviation* and *Selected Accuracy* together with the test accuracy of committee sampling (mauve) and the optimal stopping point (marked using a vertical dashed line)**

| | $acc_\Delta\%$ | $\|queries\|$ | *slope* |
|---|---|---|---|
| No Stopping | 3.00 | 671 | 4.474e-03 |
| Maximum Uncertainty Stopping | 3.40 | 661 | 5.144e-03 |
| **Standard Deviation Stopping** | **11.19** | **70** | **1.599e-01** |
| Selected Accuracy Stopping | 10.88 | 43 | 2.530e-01 |

**Table 2: Comparison of different learning slopes (spreadsheet batches, committee sampling, 1% start size)**

the optimum stopping point for a general case we recommend to use *standard deviation* as a necessary criterion and *selected accuracy* as a sufficient criterion to detect a continuous peak.
To see the amount of effort which can be reduced by AL we calculated the slopes for the stopping points as listed in Table 2. The best results were achieved with *standard deviation* stopping, resulting in an increase of the accuracy by +11.19% using a start set of just 6 spreadsheets and a training set of 70 spreadsheets compared to 671 spreadsheets of the passive classifier.

## 4.7 Discussion

The following insights can be derived from our experiments: As a first step we recommend a comparison of different classifiers. For our scenario of cell classification, we decide on a Random Forest classifier providing high accuracy and fast training time. Compared to weak-supervised approach (see Section 5) active learning requires a start set of already labeled data. However, our experiments showed, that even a very small start set consisting of 6 spreadsheets only is totally sufficient. For cell classification, we decide to use one spreadsheet per batch to reduce the annotator effort. Regarding the sampling strategies, we can conclude, that except the random approach all strategies worked well. The best slope was achieved using the *committee strategy* which is therefore our recommended one. It is however noteworthy, that this strategy requires the highest amount of computing resources, since several classifiers have to be trained in parallel. We could not identify a single stopping criterion which suffices for identifying nearly optimal stopping points. Therefore, our recommendation is a combination of a *high selected accuracy* and a *low standard deviation* of the query accuracies.

# 5 RELATED WORK

**Unlabeled Spreadsheet Corpora** We find multiple, unlabeled spreadsheet corpora in the literature: Euses [12] has served the spreadsheet community for a while. It was created with the help of search engines, issuing queries containing keywords such as "financial" and "inventory", and file type ".xls". Overall, it comprises 4,498 unique spreadsheets, organized into categories (folders) based on the used keywords. The more recent Enron corpus [13] contains 15,770 spreadsheets, extracted from the Enron email archive[1]. This corpus is unique regarding its exclusive view on the use of spreadsheets in enterprise settings. Another recent corpus is Fuse [4], which comprises 249,376 unique spreadsheets, extracted from Common Crawl[2]. Each spreadsheet is accompanied by a JSON file, which includes NLP token extraction and metrics related to the use of formulas.
So far, these three corpora have been used by researchers viewing spreadsheets from a software engineering perspective. Formula error detection and debugging [15, 26], but also usage, life-cycle, modeling, and governance of spreadsheets [1, 8, 14] are important research subjects within this community. **Table Recognition** In the literature we find various approaches with regards to layout inference and table recognition in spreadsheets as prerequisites for information extraction. These approaches can be classified as rule-based, heuristics-based, and ML-based ones. Here, we consider those, that are ML-based, like [11, 19]. More recent publications apply to some extent machine learning techniques [2, 5, 6, 19]. Other works make use of domain specific languages, such as [16], [31], and [14].
**Active Learning** The existing AL approaches can be grouped in three categories: *Membership Query Synthesis*, *Stream-Based Selective Sampling*, and *Pool-Based Sampling* [28]. Membership Query Synthesis assumes, that new data points can arbitrarily be created and the learner can request any data point from the problem space. For Stream-Based Selective Sampling a data point is being sampled from the problem space and the learner can decide, if the generated sample should be labeled or discarded. Both approaches are based on the assumption, that arbitrary real-world data points can be produced in an inexpensive way which is not the case in our scenario. Pool-Based Sampling instead assumes a fixed pool of labeled data $L$ and a large pool of unlabeled data $U$ from the problem space, which holds true for our spreadsheet dataset. Therefore, in this paper we implemented a pool-based sampling approach. In the context of spreadsheets, [6] proposed a rule-assisted AL approach to detect certain properties. The authors introduce a hybrid approach, that combines active learning with crude easy-to-write user-provided rules.
**Weak Supervision** The common notion behind weak supervision techniques is, that it is much easier to generate a large amount of so-called *weakly*-labeled data than a small amount of good-labeled data and that more data always leads to better results. Under this assumption multiple approaches have been proposed: In Snorkel [24] user defined labeling functions (LF) are used to label given data points. Writing LFs however, requires experts with domain knowledge, in contrast to spreadsheet cell labeling which can be done by non-experts also. Additionally, Snorkel users often have to provide a development set of labeled data providing quick approximate signals on the accuracies of the LFs. Hence, the human effort is not necessarily lower compared to AL.
As successor of Snorkel Snuba [32] was created, where the noisy LFs are automatically derived from a small labeled dataset. However, our experiments with the open-source code base of Snuba

[1] http://info.nuix.com/Enron.html
[2] http://commoncrawl.org/

have shown some limitations. In the current state it is not easily applicable to problems of more than two labels and it seems to be only working well for certain types of problems, to which the task of classifying spreadsheet cells does not belong to: First, our dataset is quite unbalanced, which results in LFs, that are only focusing on the majority class and second, with a growing number of labeling classes the LFs again tend to focus only on the most common label. While investigating this we discovered multiple issues in the current implementation of Snuba which prevent the overall sound concept from working well in a generalized setting.

Other methods include assigning a label per feature instead of assigning a label per data point [10], which is suitable in a setting of textual data, where it is easy to specify distinctive terms per label class, but not in the case of classifying spreadsheet cells. Another approach [21, 23] first clusters the input data and assigns then a label per cluster based on a fraction of labeled data points per cluster. Our problem space has a dimension of 159 features and early experiments showed, that it is not easy to identify appropriate clusters in such a high-dimensional space.

## 6 CONCLUSIONS

In this paper, we proposed a semi-supervised approach for cell classification, that drastically reduces the amount of training data, without any losses in accuracy. In detail, we investigated different query strategies, start set sizes, stopping criteria, and batch sizes to implement the AL cycle. With the DECO corpus, consisting of 1,165 labeled spreadsheets, we tested all these configurations and derived good settings for them. Using committee sampling, a start set size of 1% and the selected accuracy stopping criterion we are able to reduce the size of the training dataset by 90%. In absolute numbers, our AL cell classification approach obtains the same accuracies using just 76 labeled spreadsheets (6 sheets in the start set and 70 selected by the AL approach) compared to the passive classifier trained on 1,165 spreadsheets. Therefore, the effort for the human annotators was reduced from 3,029 to 197 minutes. Given this reduction of annotation work and consequently in annotation costs, it now becomes feasible to apply ML-driven cell classification and table identification in practice.

## REFERENCES


[1] Robin Abraham and Martin Erwig. 2006. Inferring templates from spreadsheets. In *Proceedings of the 28th international conference on Software engineering*. ACM, 182–191.

[2] Marco D Adelfio and Hanan Samet. [n.d.]. Schema extraction for tabular data on the web. *VLDB'16* ([n. d.]), 421–432.

[3] Yoram Baram, Ran El-Yaniv, and Kobi Luz. 2004. Online Choice of Active Learning Algorithms. *J. Mach. Learn. Res.* 5 (Dec. 2004), 255–291.

[4] Titus Barik, Kevin Lubick, Justin Smith, John Slankas, and Emerson Murphy-Hill. 2015. Fuse: a reproducible, extendable, internet-scale corpus of spreadsheets. In *MSR'15*. 486–489.

[5] Zhe Chen and Michael Cafarella. [n.d.]. Automatic web spreadsheet data extraction. In *SSQ'13*. 1.

[6] Zhe Chen, Sasha Dadiomov, Richard Wesley, Gang Xiao, Daniel Cory, Michael Cafarella, and Jock Mackinlay. 2017. Spreadsheet Property Detection With Rule-assisted Active Learning. In *CIKM '17*. ACM, New York, NY, USA, 999–1008. https://doi.org/10.1145/3132847.3132882

[7] David Cohn, Les Atlas, and Richard Ladner. 1994. Improving generalization with active learning. *Machine Learning* 15, 2 (01 May 1994), 201–221. https://doi.org/10.1007/BF00993277

[8] Jácome Cunha, João Paulo Fernandes, Jorge Mendes, and João Saraiva. 2012. MDSheet: A framework for model-driven spreadsheet engineering. In *ICSE*. IEEE Press, 1395–1398.

[9] Ido Dagan and Sean P. Engelson. 1995. Committee-based Sampling for Training Probabilistic Classifiers. In *ICML'95*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 150–157.

[10] Gregory Druck, Burr Settles, and Andrew Mccallum. 2009. Active learning by labeling features. In *In Proc. EMNLP*. ACL Press, 81–90.

[11] Julian Eberius, Christoper Werner, Maik Thiele, Katrin Braunschweig, Lars Dannecker, and Wolfgang Lehner. [n.d.]. DeExcelerator: A framework for extracting relational data from partially structured documents. In *CIKM'13*. 2477–2480.

[12] Marc Fisher and Gregg Rothermel. [n.d.]. The EUSES spreadsheet corpus: a shared resource for supporting experimentation with spreadsheet dependability mechanisms. In *SIGSOFT'05*, Vol. 30. 1–5.

[13] Felienne Hermans and Emerson Murphy-Hill. 2015. Enron's spreadsheets and related emails: A dataset and analysis. In *ICSE*. IEEE Press, 7–16.

[14] Felienne Hermans, Martin Pinzger, and Arie Van Deursen. [n.d.]. Automatically extracting class diagrams from spreadsheets. *ECOOP'10* ([n. d.]), 52–75.

[15] Felienne Hermans, Martin Pinzger, and Arie van Deursen. 2015. Detecting and refactoring code smells in spreadsheet formulas. *Empirical Software Engineering* 20, 2 (2015), 549–575.

[16] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. 2011. Wrangler: Interactive visual specification of data transformation scripts. In *SIGCHI*. ACM, 3363–3372.

[17] Elvis Koci, Maik Thiele, , Oscar Romero, and Wolfgang Lehner. 2019. A Genetic-based Search for Automatic Table Recognition in Spreadsheets. In *ICDAR'19*.

[18] Elvis Koci, Maik Thiele, Josephine Rehak, Oscar Romero, and Wolfgang Lehner. 2019. DECO: A Dataset of Annotated Spreadsheets for Layout and Table Recognition. In *ICDAR'19*.

[19] Elvis Koci, Maik Thiele, Óscar Romero Moral, and Wolfgang Lehner. 2016. A machine learning approach for layout inference in spreadsheets. In *KDIR*. SciTePress, 77–88.

[20] David D. Lewis. 1995. A Sequential Algorithm for Training Text Classifiers: Corrigendum and Additional Data. *SIGIR Forum* 29, 2 (Sept. 1995), 13–19. https://doi.org/10.1145/219587.219592

[21] Zhiliang Liu, Xiaomin Zhao, Jianxiao Zou, and Hongbing Xu. 2013. A Semi-Supervised Approach Based on k-Nearest Neighbor. *Journal of Software* 8 (04 2013). https://doi.org/10.4304/jsw.8.4.768-775

[22] 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. 2011. Scikit-learn: Machine Learning in Python. *JMLR* 12 (2011), 2825–2830.

[23] Fábio Perez, Rémi Lebret, and Karl Aberer. 2018. Cluster-Based Active Learning. *CoRR* abs/1812.11780 (2018). arXiv:1812.11780

[24] Alexander Ratner, Stephen H. Bach, Henry R. Ehrenberg, Jason Alan Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid Training Data Creation with Weak Supervision. *CoRR* abs/1711.10160 (2017). arXiv:1711.10160

[25] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. 2001. Active Hidden Markov Models for Information Extraction. In *IDA '01*. Springer-Verlag, London, UK, UK, 309–318.

[26] Thomas Schmitz, Dietmar Jannach, Birgit Hofer, Patrick W. Koch, Konstantin Schekotihin, and Franz Wotawa. 2017. A decomposition-based approach to spreadsheet testing and debugging. In *Visual Languages and Human-Centric Computing*. IEEE Computer Society, 117–121.

[27] Greg Schohn and David Cohn. 2000. Less is More: Active Learning with Support Vector Machines. *Machine Learning-International Workshop then Conference* (10 2000).

[28] Burr Settles. 2010. Active learning literature survey. *Computer Sciences Technical Report 1648* (07 2010).

[29] H. S. Seung, M. Opper, and H. Sompolinsky. 1992. Query by Committee. In *COLT '92*. ACM, New York, NY, USA, 287–294. https://doi.org/10.1145/130385.130417

[30] C. E. Shannon. 2001. A Mathematical Theory of Communication. *SIGMOBILE Mob. Comput. Commun. Rev.* 5, 1 (Jan. 2001), 3–55. https://doi.org/10.1145/584091.584093

[31] Alexey O Shigarov and Andrey A Mikhailov. 2017. Rule-based spreadsheet data transformation from arbitrary to relational tables. *Information Systems* 71 (2017), 123–136.

[32] Paroma Varma and Christopher Ré. 2018. Snuba: Automating Weak Supervision to Label Training Data. *Proc. VLDB Endow.* 12, 3 (Nov. 2018), 223–236. https://doi.org/10.14778/3291264.3291268

[33] Xinxin Wang. 1996. *Tabular abstraction, editing, and formatting*. Technical Report. University of Waretloo, Waterloo, Ontaria, Canada.

[34] Jingbo Zhu, Huizhen Wang, Eduard Hovy, and Matthew Ma. 2010. Confidence-based Stopping Criteria for Active Learning for Data Annotation. *ACM Trans. Speech Lang. Process.* 6, 3, Article 3 (April 2010), 24 pages. https://doi.org/10.1145/1753783.1753784