International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:10, No:10, 2016

Context Detection in Spreadsheets Based on Automatically Inferred Table Schema

Alexander Wachtel, Michael T. Franzen, Walter F. Tichy

***Abstract*—**Programming requires years of training. With natural language and end user development methods, programming could become available to everyone. It enables end users to program their own devices and extend the functionality of the existing system without any knowledge of programming languages. In this paper, we describe an Interactive Spreadsheet Processing Module (ISPM), a natural language interface to spreadsheets that allows users to address ranges within the spreadsheet based on inferred table schema. Using the ISPM, end users are able to search for values in the schema of the table and to address the data in spreadsheets implicitly. Furthermore, it enables them to select and sort the spreadsheet data by using natural language. ISPM uses a machine learning technique to automatically infer areas within a spreadsheet, including different kinds of headers and data ranges. Since ranges can be identified from natural language queries, the end users can query the data using natural language. During the evaluation 12 undergraduate students were asked to perform operations (sum, sort, group and select) using the system and also Excel without ISPM interface, and the time taken for task completion was compared across the two systems. Only for the selection task did users take less time in Excel (since they directly selected the cells using the mouse) than in ISPM, by using natural language for end user software engineering, to overcome the present bottleneck of professional developers.

***Keywords*—**Natural language processing, end user development; natural language interfaces, human computer interaction, data recognition, dialog systems, spreadsheet.

I. Introduction

THE main question in the End User Development (EUD) area of research is, how to allow non-programming users who have no access to source code, to program a computer system or extend the functionality of an existing system [1]. Tables have been used for at least 7,000 years [2]. Spreadsheet programs such as Microsoft Excel have become ubiquitous. The created spreadsheets are not only the traditional tabular representation of relational data that convey information space efficiently, but also allow a continuous revision and formula-based data manipulation. It is estimated that each year hundreds of millions of spreadsheets are created [3]. Myers [4] and also Scaffidi [5] compared the number of end users and professional programmers in the United States. Nearly 90 million people use computers at work and 50 million of them use spreadsheets. In a self-assessment 12 million considered themselves as programmers, but only 3 million people are professional programmers.

In 1979, Ballard et al. [6]-[8] introduced the Natural Language Computer (NLC) that enables end users to program simple arithmetic calculations using natural language. In 2015, Wachtel [9] presented our first prototype of an assistant system that uses natural language understanding and a dialog management system to allow inexperienced users to manipulate spreadsheets with natural language. Motivated by a pilot study based on the selected problems from Frey's book Microsoft Excel 2013 [10] the system requests missing information and is able to resolve ambiguities by providing alternatives to choose from. Furthermore, the dialog system must resolve references to previous results, allowing the construction of complex expressions step-by-step. In early 2016, Wachtel [11] extended the prototype with an active ontology. The idea of active ontology was first presented in 2006 by Guzzoni [12]. In general, an ontology is a formal representation of knowledge. By adding a rule evaluation system, a fact store and sensor nodes to an ontology it becomes an execution environment rather than just a formal representation of knowledge. Sensor nodes register certain events and save them in the fact store. An evaluation mechanism tests the new facts against the existing rules and performs the associated actions if one or more rules apply.

In this work, the natural language dialog system has been extended with a machine learning component. It synthesizes formulas without explicit cell references. First, the rows of a spreadsheet are divided into different classes and the table's schema is made searchable for the dialog system. In the case of user input, it searches for headers, data values from the table and key phrases for operations. Implicit cell references like people of age 18 are then resolved to explicit references using the schema. 88 cell characteristics were automatically extracted in a body of 2145 tables, and reduced to 42 cell characteristics by the principal component analysis. The classification was carried out using a Conditional Random Field [13], probabilistic graphical models for sequence labeling. The method was trained on 4,000 lines and evaluated on about 400 lines. Overall, we evaluate the following research questions: (RQ1) *how well can a structure of the table be detected in a spreadsheet?* and (RQ2) *how well can unrestricted natural language of an end user be mapped to a table structure?*

II. Data Recognition in Spreadsheets

To identify the cognitive processes that occur during the

Alexander Wachtel is with the Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany (corresponding author, phone: +49-721-60846320; fax: +49-721-608-47343; e-mail: Alexander.Wachtel@kit.edu).

Michael T. Franzen is with the Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany (e-mail: Michael.Franzen@student.kit.edu).

Walter F. Tichy is professor and chair of programming systems with the Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany (e-mail: Walter.Tichy@kit.edu).

reading of a table, we rely on the work of Wang [14]. In 1996, she describes the most complete model of tables and formulates the cognitive processes of a reader as (1) an understanding of the process to recognize the logical structure of the table, (2) a search process to track relevant information and (3) to answer an interpretation and comparison process to questions of the reader (See Fig. 1). More precise investigations in the field of cognitive psychology [15] allow us to define the state transitions between the stages (a) target specification, (b) category selection, (c) information extraction, and (d) recycling. Finally, the process of the reader can be modeled as a state machine (See Fig. 1).

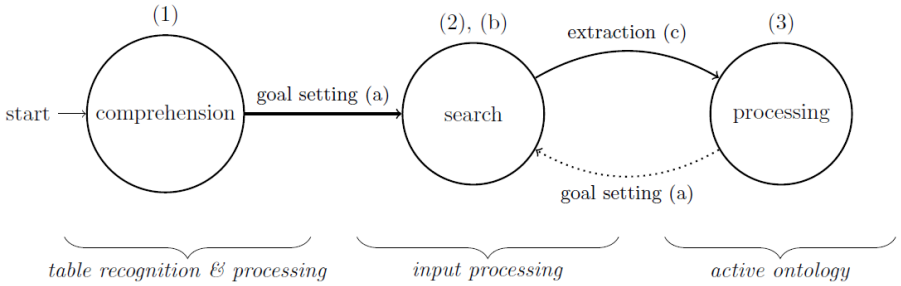

Fig. 1 Cognitive stages of an end user in ISPM. Stages (1)-(3) from [14]; processes (a)-(d) from [15]

A. Preprocessing

Before any user queries, the given spreadsheets are preprocessed to capture their tabular regions. Subsequently, an efficiently searchable and alterable data structure, an *abstract table*, as defined by [14] is constructed for each of the tables. An *abstract table* can be defined as a set of labeled domains with an access function δ. A *labeled set* is a set together with a label, each corresponding to one of the table's categories such as *name* or *age* in Fig. 2.

	A	B	C	
1	**Table 1: persons**			CAPTION
2	**name**			SUPER HEADER
3	**first name**	**last name**	**age**	HEADER
4	**group A**			GROUP HEADER
5	Sloane	Morgan	37	DATA
6	Dustin	Brewer	33	DATA
7	Valentine	Yates	38	DATA
8	Michael	Gregori	50	DATA
9	**group B**			GROUP HEADER
10	Ina	Hoffman	40	DATA
11	Oliver	Hopkins	27	DATA
12	Damon	Vasquez	22	DATA
13	Mark	Richards	25	DATA

Fig. 2 A table annotated with row labels. The super header name is subdivided into the canonical headers first name and last name. Each person is part of either group A or group B which is depicted as group headers. Note: It depicts a simple table and is for illustration purposes only. A detailed evaluation on the accuracy of table hierarchies is however not feasible because there seems to be no real ground-truth (See [16] for a study on this topic)

While printed tables sometimes contain a *stub box* representing the primary keys of tabular data as vertically stacked categories, spreadsheets often omit this feature. Instead, the rows of the table can be regarded as a tabular category and thus as a labeled domain itself.

A labeled domain is either the labeled empty set or a labeled set of uniquely labeled sets. This definition induces a tree-like label hierarchy illustrated in Fig. 3.

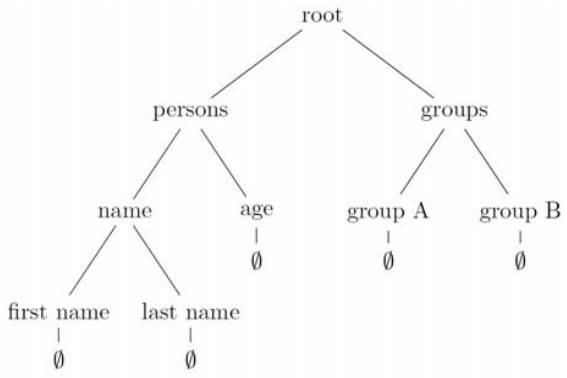

Fig. 3 The label hierarchy of the table depicted in Fig. 2

Finally, the *access function δ* maps a *label* to its corresponding tabular regions. For instance, δ (first name) = {A5; …; A13} as illustrated in Fig. 4.

Given an abstract table, natural language references to tabular regions such as "the age of people in group A" could be resolved to δ (age) ∩ δ (group A) = {A4, A5, A6, A7} and forwarded to the active ontology. To infer these data structures, we adopt a method first proposed by Pinto et al. [17], which was recently applied [18], [19] to both table recognition and table processing. Pinto et al. showed that linear-chain Conditional Random Fields (CRFs) [13] accurately predict a table's row classes. CRFs are probabilistic graphical models for sequence labelling. The goal is to assign a label to each item in a sequence. To achieve this, features are computed for each item. Afterwards, the dependencies across items in a single sequence are inferred from the training data. Using these factors, a probability distribution is constructed

which is used to infer the optimal label sequence from a feature sequence. We use the following row labels to identify and construct a spreadsheet's abstract table objects (Table I).

Given such a label sequence, we split the spreadsheet's rows into vertically stacked tabular regions similar to [19]. A table starts with either a caption, a super header or a canonical header. A table ends whenever a data, a non-relational or a group header row is followed by a caption, a super header or a canonical header. An abstract table is implemented consisting of three separate data structure: A logical tree corresponding to the label hierarchy as illustrated in Fig. 3, a physical tree corresponding to the access function δ (See Fig. 4) and a data index. To construct those data structures, we parse the row label sequence in a recursive manner. The algorithm parses one row at a time. In case of a caption row, a new node in the logical structure (*persons*) is created and assigned the entire tabular region below the caption (A2 to C13). The algorithm then proceeds by investigating the row below the caption. For super header and canonical header rows, the algorithm continues with each range of merged columns in the currently investigated area. Group headers are parsed distinct from the main recursion. They horizontally span the entire table and are vertically partitioned using the given label sequence.

TABLE I
DESCRIPTION OF THE ROW LABELS

Caption	superordinates the entire table
Super Header	superordinates another super header row or a canonical header row
Canonical Header	directly superordinates the data
Group Header	partitions the data region
Data	contains the data entries
Aggregate	contains some aggregation of the data entries such as the sum of each column
Non-Relational	describes all rows that do not meet one of the above requirements

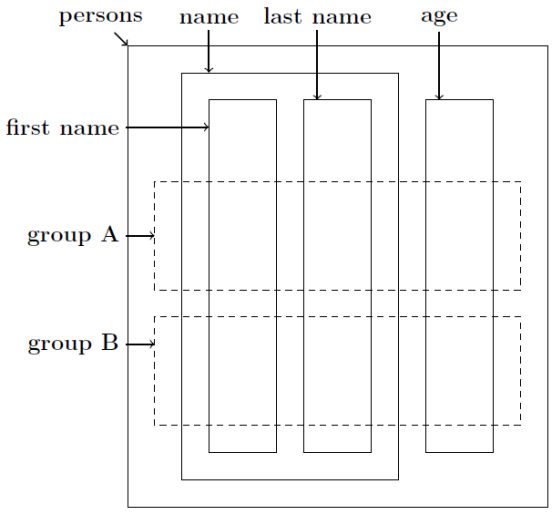

Fig. 4 Tabular regions of the headings in Fig. 2

B. Input Processing

Given the logical structure of a table and a natural language user query, we resolve natural language cell references by first matching the table data and header labels with the words of the user input. After stripping the user request from special characters and normalizing it to lower case.

We distinguish between four cases:

- When the user input consists of a single header label L, we resolve the sentence to the tabular region δ (L). This applies to a user input *What is the average age?* where L = age.
- When the user input consists of two header labels L_1 and L_2, it is resolved to $\delta(L_1) \cap \delta(L_2)$. This applies to a user input *What is the average age of people in group A?* where L_1= age and L_2 = group A.
- When the user input consists of a single header label L and a data value v found in lines $M = m_1, \ldots, m_n$, we resolve the natural language query to $\delta(L) \cap \delta(v \mid v \in M = \{m_1, \ldots, m_n\}$. This applies to a user input like *What is the age of Dustin?* where L = age and v = Dustin.
- Finally, when the user input consists of two header labels L_1 and L_2 in that order, as well as a single data value v. This works analogue to case #3 with the exception that the data entry is required to be in the region of header L_2. This case gives end users the opportunity to avoid ambiguities. It applies to inputs like *What is the average age of people whose name is Dustin?*

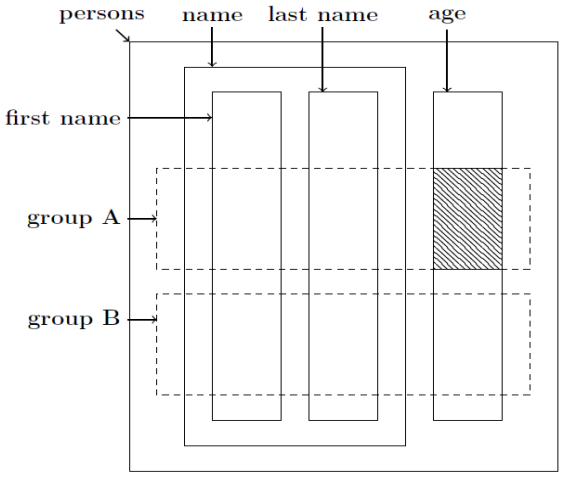

Fig. 5 Relevant cells of the user input sum of the age of people in group A

Words are matched as 1- and 2-grams of the input sequence. In the case of nouns, both the singular and plural form are tried. A simple keyword matching procedure is used to identify relevant operations such as average or sort. If no operation keyword is found, it is assumed that the user wants to select the relevant data cells. Given the request operation and the relevant cells, a new input with explicit cell references is generated which is used by our active ontology to synthesize the output. E.g., given an input sum the age of

people in group A, we first identify the relevant cells δ (age) ∩ δ (group A) and get the formula sum C5, C6, C7, C8 (See Fig. 5). This result is then forwarded to an active ontology which synthesizes the resulting Excel program SUM(SUM(SUM(C5, C6), C7), C8), as described in [11].

C. Data Sets

We downloaded 1,177 files from the WEB corpus [18], which consists of a precompiled list of roughly 410,000 .xls-files, originally found in the WebClue09 dataset [20]. Of those files 21:33% were non-relational, 14.53% were forms, 5.27% had non-English components and 1.27% were corrupt. The remaining files of the stripped corpus contained 2,145 manually extracted tables, totaling 39,0657 rows and 25,586 columns (See Table II).

Of the approximately seven million cells in the entire data set, we automatically extracted 88 layouts and features such as the cell color, text color or whether the content is numerical or textual. To reduce the feature dimensionality, we eliminated 26 features which were never used and applied a Principal Component Analysis (PCA) [21] on the remaining 62 features. We reduced the dimensionality along the 15 principal component axes. To train the linear-chain CRF, we constructed row features by averaging the 15 features of each cell of a row. The mean values were subsequently binarized with a threshold of zero (since the PCA was applied on standardized feature vectors, the expected value is zero, see Table III).

TABLE II
ROW LABELS

Average number per table	
rows	182
columns	13
cell	3226
Total	
rows	390657
columns	25586
cell	6920548

TABLE III
ROW LABELS

	Variable	min	max	μ	σ
1	font color - bleu	0	1	0.42	0.49
2	font size	0	0.13	0.02	0.02
	…				
87	content - text	0	1	0.21	0.41
88	content - numeric	0	1	0.26	0.44

III. EVALUATION

We implemented the algorithms specified in Section II-A and II-B. CRF was trained and tested on approximately 4,000 and 400 rows respectively. The results in Table IV show the varying precision, recall and F1 values for each of the tested rows. It is noticeable, especially in context to previous work, that the group and super header rows are misclassified fairly often. This may be the result of an incomplete, low-level feature set. Adelfio et al. [18] for example used a hand-selected feature set.

The overall distribution of row classes (five captions, eight super headers, six group headers, 16 canonical headers, 344 data rows, five aggregate rows and 40 non-relational rows) yields a precision, recall and F1-score of 93.4%, 94.1% and 93.8%, respectively.

TABLE IV
RECOGNITION RATES OF ROW CLASSES OF THE CONDITIONAL RANDOM FIELD

	Precision	Yield	F1
VCAPTION	80.00 %	80.00 %	80.00 %
VDATA	98.24 %	97.38 %	97.81 %
VGROUPHEADER	100 %	16.66 %	28.56 %
VHEADER	66.66 %	87,50 %	75.67 %
VNONREL	75.00 %	97.5 %	84.78 %
VSUPERHEADER	100 %	37.50 %	54.55 %

A. Input Processing

We performed a user study with twelve German students, nive females and 3 males, eleven computer science and one biology undergraduates, to evaluate the input processing algorithms. In a self-assessment, 8% considered themselves as experts, 50% as advanced users, and 42% as beginners regarding Excel spreadsheets.

Given three different tables of simple structure (canonical), we asked each user to perform four different tasks:

- **sort**: in a given spreadsheet with 50 entries of companies including name, mail address, website, and city, the participants should sort the entries by the company's mail address.
- **group**: in a given spreadsheet with 50 entries of credit cards including provider name, person name, and card number, the participants should group the entries by the provider name.
- **select**: in a given spreadsheet with 50 entries of persons including name, age, job, and academic grad, the participants should identify all persons who are 18 years old.
- **aggregate**: in the same table as the select use case, the participants should calculate the age of all persons who have Dr. as their academic title.

TABLE V
EVALUATION RESULT OF THE DIALOG SYSTEM

Total	
Number of Human-Computer Interaction	360
Successfully Solved	78 %
Objectives	
Successful Sorting	88 %
Successful Grouping	75 %
Successful Selection	88 %
Successful Aggregation	63 %

Each task was performed twice by each user. Once, they were asked to only use our system and once to only use the native Excel environment. We measured the required time for each variation of a task. Furthermore, we asked the users to

rate how satisfying their solution and how helpful ISPM was. Both the order of the tasks and the order of a task's variation were randomly chosen. As seen in Table II the system successfully solved about 78% of the given tasks.

Regarding the experimental design, the study was conducted to compare the native Excel environment to Excel Add-in we developed in regards to usability. The participants performed each task twice. However, they used only Excel in the first run and only the natural language dialog system in the second run. Each of these two variations were performed exactly once by every participant. The experiment was not designed to investigate any learning effects.

We asked participants for their English proficiency on a five-point scale and how often they use Excel. After each variation of a task, a user was asked to rate his satisfaction with the solution from 0 (*not satisfied*) to 4 (*totally satisfied*). After both variations of a task were completed, a user was asked to rate the helpfulness of our system on a scale from 0 (*not helpful*) to 4 (*very helpful*). The results are shown in Table VI. It is noticeable our system outperforms the native Excel environment regarding the mean subjective satisfaction of a user. The helpfulness was rated high ($\mu \geq 3.25$) on all tested tasks.

TABLE VI
THE MEAN AND STANDARD DEVIATION OF THE VARIABLES

Variable	μ	σ
Satisfaction - *aggregate*, Excel	2.75	1.42
Satisfaction - *aggregate*, ISMP	3.42	1.24
Satisfaction - *group*, Excel	2.17	1.85
Satisfaction - *group*, ISMP	3.92	0.29
Satisfaction - *select*, Excel	3	1.13
Satisfaction - *select*, ISMP	3.83	0.39
Satisfaction - *sort*, Excel	3.17	1.4
Satisfaction - *sort*, ISMP	3.67	1.15
Helpfulness - *aggregate*	3.75	0.62
Helpfulness - *group*	3.83	0.39
Helpfulness - *select*	3.83	0.39
Helpfulness - *sort*	3.25	1.29

Furthermore, we measured the time until completion of each task variation (See Fig. 6). Only 25% had solved the group task after 110 s. 75% of the participants solved that task in our system. Without ISPM, after 175s about 50% and after 310s all participants finished the group task. Analogous conclusions can be drawn during the sorting and aggregation. The only exception is the selection task. End users resolved that task with the mouse, and were faster without ISPM.

Finally, we tried to answer the given research questions, (RQ1) how well can a structure of the table be detected in a spreadsheet? And, (RQ2) how well can unrestricted natural language of an end user be mapped to a table structure?

- RQ1: the detection of the table structure was good, but is still limited. In our case, only row-oriented tables can be handled. However, if more tables are found in spreadsheet, the dialogue systems ask users which table they want to work on.
- RQ2: in our case the unrestricted natural language can be mapped to the table structure. However, the system responds only to answers that contain some index of the table (*first*, *the second one*, etc.).

IV. Related Work

Our work combines different research on data recognition and manipulation in spreadsheets, end user programming, programming in natural language, and natural language dialog systems.

A. Data Recognition and Manipulation in Spreadsheets

Lopresti [22] presented a tabular survey of automated table processing in 1999. Also, Embley [23] generalized paradigms for table processing in a research survey in 2006. Research on table composition and table analysis has improved the understanding of the distinction between the logical and physical structures of tables, and has led to improved formalisms for modeling tables. Adelfio [18] extracts schemas for tabular data on the web. The structure of these tables is not accessible to the web crawlers because the schemas are not explicitly stored as table metadata. The schemas of these data tables are determined using a classification technique based on Conditional Random Fields in combination with a novel feature encoding method called logarithmic binning, which is specifically designed for the data table extraction task. In 2001, Hu [16] presented a detailed analysis of why table ground-truthing is so hard, including the notions that there may exist more than one acceptable truth and/or incomplete or partial truths. While understanding natural language is difficult, tables and other structured information make it easier to interpret new items and relations, Tijerno et al. [24] introduced an approach to generating ontologies based on table analysis. Based on conceptual modeling extraction techniques, the approach attempts to (i) understand a table's structure and conceptual content; (ii) discover the constraints that hold between concepts extracted from the table; (iii) match the recognized concepts with ones from a more general specification of related concepts; and, (iv) merge the resulting structure with other similar knowledge representations. The project Senbazuru [25], [26] from the University of Michigan deals with the semi-automatic extraction of relational data from spreadsheets. The aim of the application is to simplify the integration of spreadsheets in relational databases and provide users assistance. Tables are also searched by Conditional Random Field [13]. Instead to raise the relevant characteristics and generate unigram and bigram features quantitatively, as has been already shown in this work, they use 18 line features from which they generate unigram and bigram models. NLyze [27], an Add-In for Microsoft Excel that has been developed by Gulwani at the same time as our system. It enables end users to manipulate spreadsheet data by using natural language. It uses a separate domain-specific language for logical interpretation of the user input. Instead of recognizing the tables automatically, it uses canonical tables which should be marked by the end user. Another of Gulwani's tool QuickCode [28] deals with the production of

the program code in spreadsheets through input-output examples provided by the end user [29]. It automates string processing in spreadsheets using input-output examples and splits the manipulations in spreadsheet by entering examples. The focus of his work is on the synthesizing of programs that consist of text operations.

B. End User Programming

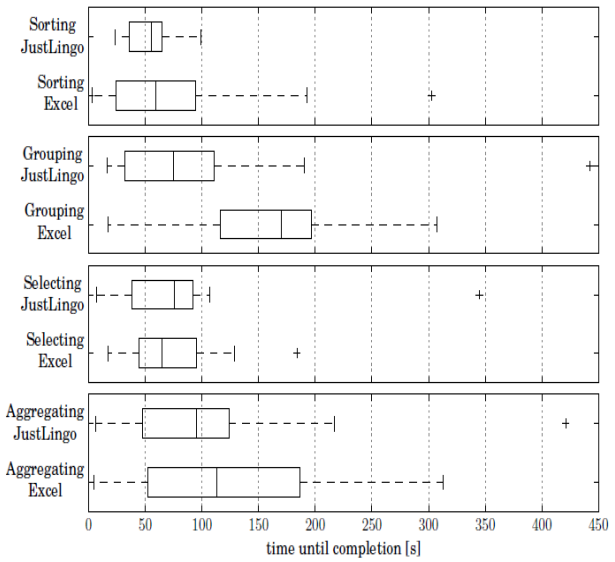

Fig. 6 Box plot of required solution time in both our system

Paternò [30] introduces the motivations behind end user programming defined by Liberman [1] and discusses its basic concepts, and reviews the current state of art. Various approaches are discussed and classified in terms of their main features and the technologies and platforms for which they have been developed. In 2006, Myers [4] provides an overview of the research in the area of End User Programming. As he summarized, many different systems for EUD have already been realized [31], [32], [29]. However, there is no system such as our prototype that can be controlled with natural language. During a study in 2006, Ko [31] identifies six learning barriers in End User Programming: design, selection, coordination, use, understanding and information barriers. In 2008, Dorner [33] describes and classifies EUD approaches taken from the literature, which are suitable approaches for different groups of end users. Implementing the right mixture of these approaches leads to embedded design environments, having a gentle slope of complexity. Such environments enable differently skilled end users to perform system adaptations on their own. Sestoft [34] increases expressiveness and emphasizing execution speed of the functions thus defined by supporting recursive and higher order functions, and fast execution by a careful choice of data representation and compiler technology. Cunha [35] realizes techniques for model-driven spreadsheet engineering that employs bidirectional transformations to maintain spreadsheet models and synchronized instances. Begel [36] introduces voice recognition to the software development process. His approach uses program analysis to dictate code in natural language, thereby enabling the creation of a program editor that supports voice-based programming.

The idea of programming in natural language was first proposed by Sammet [37], but enormous difficulties have resulted in disappointingly slow progress. One of the difficulties is that natural language programming requires a domain-aware counterpart that asks for clarification, thereby overcoming the chief disadvantages of natural language, namely ambiguity and imprecision. In recent years, significant advances in natural language techniques have been made, leading, for instance, to IBM's Watson [38] computer winning against the two Jeopardy! world champions, Apple's Siri routinely answering wide-ranging, spoken queries, and automated translation services such as Google's becoming usable [39], [40]. In 1979, Ballard et al. [6]-[8] introduced their Natural Language Computer (NLC) that enables users to program simple arithmetic calculations using natural language. Our prototype extends the idea with a dialog system component for step-by-step construction of complex expression and enables users to perform tasks they otherwise would not be able to accomplish. Although NLC resolves references as well, there is no dialog system. Metafor introduced by Liu et al. [41] has a different orientation. Based on user stories, the system tries to derive program structures to support software design. A different approach regarding software design via natural language is taken by RECAA [42]. RECAA can automatically derive UML models from the text and also keep model and specification consistent through an automatic feedback component. A limited domain end-to-end programming is introduced by Le. SmartSynth [43] allows synthesizing smartphone automation scripts from natural language description. However, there is no dialog interaction besides the results output and error messages.

C. Natural Language Dialog Systems

Many dialog systems have already been developed. Commercially successful systems, such as Apple's Siri, actually based on active ontology [12], and Google's Voice Search [44], [45] cover many domains. Reference resolution makes the systems act natural. However, there is no dialog interaction. The Mercury system [46] designed by the MIT research group is a telephone hotline for automated booking of airline tickets. Mercury guides the user through a mixed initiative dialog towards the selection of a suitable flight based on date, time and preferred airline. Furthermore, Allen [47] describes a system called PLOW. As a collaborative task agent PLOW can learn to perform certain tasks, such as extracting specific information from the internet, by demonstration, explanation, and dialog.

V. Conclusion

Since their invention, digital computers have been programmed using specialized, artificial notations, called programming languages. However, only a tiny fraction of human computer users can actually work with those notations. With natural language and EUD methods, programming

would become available to everyone and enable end users to program their systems or extend their functionality without any knowledge of programming languages. It is a challenge how to connect natural language of an end user to the data in the spreadsheets and to use machine learning techniques for the context interpretation within spreadsheets. It could be achieved by harnessing the similarities and differences of nearby table rows through the use of a novel set of features and a feature processing scheme [18].

We presented the machine learning module for the natural language dialog system that synthesizes spreadsheet formulas without explicit cell references. This module acted as preprocessing step. First, the rows of a spreadsheet are divided into different classes using a conditional random field [13]. The result is the construction of logical and physical structures of tables in a spreadsheet. The dialog system can search for values in the schema of the table and it allows users to manipulate spreadsheet data by using natural language. Ordinary, natural language would enable almost anyone to program and would thus cause a fundamental shift in the way computers are used. Rather than being a mere consumer of programs written by others, each user could write his or her own programs [48]. However, programming in natural language remains an open challenge [40].

VI. Future Work

To enable the system for End User Programming, ISPM should be extended for implementation of Excel scripts called macros from natural language input. We are also exploring ways to extend the system functionality with the help of the dialog. The system needs to be extended for handling graphs and charts, and supporting loops. Furthermore, there are some properties of tables, which are not considered in the current system and can potentially lead to problems.

Integration of knowledge databases: natural language queries contain not only words that exist within a table. Given the columns name and age, the following two statements are equivalent: *what is the age of Alex?* And, *how old is Alex?* During the first request would be processed correctly, but the second statement would not work. An integration of thesauri (as WordNet [49], [50]) could significantly improve the processing of such requests. Furthermore, knowledge databases (as Yago [51] or freebase [49]) can be beneficial [52], because sufficient headers are present in a table. Using an integrated knowledge base might be a Named Entity Recognition on the table (instead of just be applied to the request) and to deduce the missing information.

Resolution of named entity disambiguation: ambiguities in natural language queries could be handled by the active dialog system. If there are several candidates for answer a request or a request was not understood, the system would ask the user for clarification. For example, consider a table with two entries named Michael, but different place of birth (e.g. London and Berlin) and age (e.g. 25 years and 28 years old). If the user asks the system: *What is the age of Michael*? The currently implemented method returns both entries (25 and 28). It should response with the question *Do you mean the one born in London or the one born in Berlin?*

Dynamic operations and larger instruction set: while forwarding the explicit cell references in our system is static, it would be desirable if all commands are based on the AO, dynamically supported. Such a system would use the pattern of active ontology and bring the dependency trees of the Stanford parser in conjunction to support the correct parameter order of an operation.

References

[1] H. Liberman, “End-User Development: An Emerging Paradigm,” 2006.
[2] M. Hurst, “The interpretation of tables in texts,” University of Ediburgh, Ph.D., 2000.
[3] R. Abraham, “Header and Unit Inference for Spreadsheets Through Spatial Analyses,” in IEEE Symposium on Visual Languages – Human Centric Computing, 2004.
[4] B. A. Myers, “Invited Research: Overview End-User Programming,” CHI, 2006.
[5] B. M. Christopher Scaffidi, Mary Shaw, “Estimating the numbers of end users and end user programmers,” in Proceedings of the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing, ser. VLHCC ’05. IEEE Computer Society, 2005.
[6] B. Ballard, “Programming in natural language: NLC as a prototype,” Association for Computing Machinery (ACM), 1979.
[7] A. Biermann, “Toward Natural Language Computation,” American Journal of Computational Linguistics, 1980.
[8] “An experimental study of natural language programming,” Int. J. Man-Machine Studies, 1983.
[9] A. Wachtel, “Initial implementation of natural language turn-based dialog system,” International Conference on Intelligent Human Computer Interaction (IHCI), 2015.
[10] C. D. Frye, “Microsoft Excel 2013, Step by Step,” O’Reilly Media, 2013.
[11] A. Wachtel, “A Natural Language Dialog System Based on Active Ontologies,” Proceedings of the Ninth International Conference on Advances in Computer-Human Interactions, 2016.
[12] D. Guzzoni, “Active: A unified platform for building intelligent web interaction assistants,” in Web Intelligence and Intelligent Agent Technology Workshops, 2006. WI-IAT 2006 Workshops. 2006 IEEE/WIC/ACM International Conference on. IEEE, 2006, pp. 417–420.
[13] J. Lafferty, A. McCallum, and F. C. Pereira, “Conditional random fields: Probabilistic models for segmenting and labeling sequence data,” 2001.
[14] X. Wang and D. Wood, Tabular abstraction, editing, and formatting. Citeseer, 1996.
[15] J. Guthrie, “Literacy as multidimensional: Locating information and reading comprehension,” in Educational Psychologist, 22, 1987.
[16] J. Hu, “Why table ground-truthing is hard,” in International Conference on Document Analysis and Recognition, 2001.
[17] 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 informaion retrieval. ACM, 2003, pp. 235–242.
[18] M. D. Adelfio and H. Samet, “Schema extraction for tabular data on the web,” Proceedings of the VLDB Endowment, vol. 6, no. 6, pp. 421–432, 2013.
[19] Z. Chen and M. Cafarella, “Automatic web spreadsheet data extraction,” in Proceedings of the 3rd International Workshop on Semantic Search over the Web. ACM, 2013, p. 1.
[20] J. Callan, M. Hoy, C. Yoo, and L. Zhao, “Clueweb09 data set,” 2009.
[21] I. Jolliffe, Principal component analysis. Wiley Online Library, 2002.
[22] D. Lopresti, “A tabular survey of automated table processing,” in GREC, 1999.
[23] D. Embley, “Table-processing paradigms: A research survey,” in International Journal of Document Analysis, 2006.
[24] Y. Tijerno, “Towards ontology generation from tables,” in Springer Science, 2005.
[25] Z. Chen, “Automatic web spreadsheet data extraction,” in Proceedings of the 3rd International Workshop on Semantic Search over the Web ACM, 2013.

[26] "Senbazuru: A prototype spreadsheet database management system," in VLDB Endowment 6, 2013.
[27] S. Gulwani, "NLyze: Interactive programming by natural language for spreadsheet data analysis and manipulation," SIGMOD, 2014.
[28] "Automating string processing in spreadsheets using input-output examples," in ACM SIGPLAN, 2011.
[29] A. Cypher, "Watch what I do: programming by demonstration," in MIT Press, 1993.
[30] F. Paternò, "End user development: Survey of an emerging field for empowering people," in ISRN Software Engineering, vol. 2013, 2013.
[31] A. Ko, "Designing the Whyline: A Debugging Interface for Asking Questions About Program Failures," in CHI, 2004.
[32] S. Gulwani, "Spreadsheet data manipulation using examples," in ACM, 2012.
[33] V. W. Christian Dorner, Michael Spahn, "End user development: Approaches towards a flexible software design," in Proceedings of the European Conference on Information Systems, 2008.
[34] P. Sestoft, "Sheet-defined functions: Implementation and initial evaluation," 2013.
[35] J. Cunha, "Bidirectional Transformation of Model-Driven Spreadsheets," Springer Lecture Notes in Computer Science, 2012.
[36] A. Begel, "Spoken Language Support for Software Development," Ph.D. Thesis, Berkeley, 2005.
[37] J. E. Sammet, "The Use of English as a Programming Language," Communication of the ACM, March 1966.
[38] D. Ferrucci, "Building Watson: An Overview of the DeepQA Project," Association for the Advancement of Artificial Intelligence, 2010.
[39] H. Liu, "Toward a programmatic semantics of natural language," Visual Languages and Human Centric Computing, 2004.
[40] C. L. Ortiz, "The Road to Natural Conversational Speech Interfaces," IEEE Internet Computing, March 2014.
[41] H. Liu, "Metafor: Visualizing stories as code," 10th international conference on Intelligent user interfaces, 2005.
[42] S. Körner, "Transferring Research Into the Real World - How to Improve RE with AI in the Automotive Industry," 2014.
[43] V. Le, "SmartSynth: Synthesizing Smartphone Automation Scripts from Natural Language," MobiSys, 2013.
[44] J. R. Bellegarda, "Spoken Language Understanding for Natural Interaction: The Siri Experience," Springer New York, 2014.
[45] J. D. Williams, "Spoken dialogue systems: challenges and opportunities for research," 2009.
[46] S. Seneff, "Response planning and generation in the MERCURY flight reservation system," 2002.
[47] J. Allen, "PLOW: A Collaborative Task Learning Agent," Association for the Advancement of Artificial Intelligence, 2007.
[48] W. F. Tichy, "Universal Programmability - How AI Can Help. Artificial Intelligence Synergies in Software Engineering," May 2013.
[49] K. Bollacker, "Freebase: a collaboratively created graph database for structuring human knowledge," in ACM SIGMOD, 2008.
[50] A. Budanitsky, "Semantic distance in wordnet: An experimental, application-oriented evaluation of five measures," in Workshop on WordNet and Other Lexical Resources, 2001.
[51] F. Mahdisoltani, "Yago3: A knowledge base from multilingual wikipedias," in 7th Biennial Conference on Innovative Data Systems Research CIDR 2015, 2015.
[52] G. Limaye, "Annotating and searching web tables using entities, types and relationships," in VLDB Endowment Bd. 3, 2010.