Facilitating Document Reading by Linking Text and Tables

Dae Hyun Kim[1], **Enamul Hoque**[1], **Juho Kim**[2], **and Maneesh Agrawala**[1]

Stanford University
Stanford, CA, USA
{dhkim16, enamul, maneesh}@cs.stanford.edu

KAIST
Daejeon, Republic of Korea
juhokim@kaist.ac.kr

Information's Impact *Successful searchers who say online health information…*	%
Affected a decision about how to treat an illness or condition	44
Led them to ask a doctor new questions or get a second opinion	38
Changed approach to maintaining own health or health of someone they care for	34
Changed the way they think about diet, exercise, and stress	30
Changed the way they cope with a chronic condition or manage pain	25
Affected a decision about whether to see a doctor or not	17

Of the successful searchers, 44% said the information they found online affected a decision about how to treat an illness or cope with a medical condition. Again, there was no significant difference between those who searched on behalf of someone else and

Men are more involved with internet connections than women.		
Traits	**% of online men**	**% of online women**
Those who go online at work and have hi-speed connections at work	75*	59
Dial-up users at home who would like to have hi-speed connections at home	47*	34
Those who go online at home and have hi-speed connections at home	46*	39
Dial-up users not aware of hi-speed availability at home	22	30*
Those who go online at work and don't know about connections at work	10	22*
Those who go online at home and don't know about connections at home	2	3

Further, women were not as interested as men in making an upgrade from slow to fast connections: Of those with dial-up connections, significantly more men than women said they were interested in getting high speed connections.

Figure 1. Documents often include tables that provide evidence for arguments made in the main body text. In explicit references (left), the sentence text *"Of the successful searchers, 44% said the information they found online affected a decision about how to treat an illness or cope with a medical condition"* directly matches the text and numbers in the table cells (yellow highlights). In implicit references (right), the sentence text *"... Of those with dial-up connections, significantly more men than women said they were interested in getting high speed connections"* corresponds to row and column headers and readers must identify data cells at the intersection of two – i.e. the cells containing 47 and 34. Our interactive document reader automatically extracts such references for an input PDF document. Readers can click on a sentence to highlight the corresponding table cells and vice versa.

ABSTRACT

Document authors commonly use tables to support arguments presented in the text. But, because tables are usually separate from the main body text, readers must split their attention between different parts of the document. We present an interactive document reader that automatically links document text with corresponding table cells. Readers can select a sentence (or tables cells) and our reader highlights the relevant table cells (or sentences). We provide an automatic pipeline for extracting such references between sentence text and table cells for existing PDF documents that combines structural analysis of tables with natural language processing and rule-based matching. On a test corpus of 330 (sentence, table) pairs, our pipeline correctly extracts 48.8% of the references. An additional 30.5% contain only false negative (FN) errors – the reference is missing table cells. The remaining 20.7% contain false positive (FP) errors – the reference includes extraneous table cells and could therefore mislead readers. A user study finds that despite such errors, our interactive document reader helps readers match sentences with corresponding table cells more accurately and quickly than a baseline document reader.

UIST '18, October 14–17, 2018, Berlin, Germany

 ISBN 978-1-4503-5948-1/18/10…$15.00

DOI: https://doi.org/10.1145/3242587.3242617

ACM Classification Keywords

Human-centered computing: User interface design; Computing methodologies: Natural language processing

Author Keywords

Interactive documents; Visualization; Text analysis.

INTRODUCTION

Data tables frequently appear in news articles, financial reports and scientific articles. For example, a news article may describe a trend, a relationship or a comparison in the text, and include a table that provides additional corroborating data. Fully understanding the document often requires mentally connecting and making sense of the text together with the corresponding table. In fact, previous work has shown that people can achieve much higher recall by jointly reading the text and tables in a journal article than by looking at the tables or the surrounding text alone [17].

Unfortunately, reading text together with a data table is challenging. As shown in Figure 1, the body text can contain explicit references (left), where the sentence text directly matches text in table cells or implicit references (right), where the sentence text matches the text in row and column header cells, but leaves it up to readers to identify the data cells at the intersection of the two. Moreover, readers must split their attention between the text and table and mentally integrate the two mutually dependent information sources. Such split-attention increases cognitive load [8, 15]. As a consequence, people often struggle to associate the text with the corresponding cells in the table, especially if the table is large and the text

references multiple cells. Moreover, readers must break their flow and move their locus of attention from the main body text to the table and back again. The more time it takes to find the corresponding table cells, the more difficult it is to smoothly resume reading the main body text. Although text references and corresponding table cells are intended to be read together, many readers end up trying to make sense of them separately.

We present an interactive document reader designed to facilitate reading such documents and reduce split attention. Readers can select a sentence (or table cells) and our reader highlights the corresponding table cells (or sentences). We provide an automatic pipeline for extracting such references between body text and table cells for an input PDF document. After breaking the document into sentences and tables, our pipeline considers each (sentence,table) pair and operates in three main stages: (1) In the table structure extraction stage, it identifies the data type (e.g. text, number, percent, money, etc.) and cell type (title, header, or data) of each table cell. (2) It next matches sentence text to cells based on natural language processing (NLP) techniques. (3) Finally, it applies rule-based refinement of the matches based on the table structure. For each sentence, the pipeline either outputs a reference consisting of one or more matching cells in the table, or it outputs a null reference if it cannot find such a match.

We compare our automatically generated references to human-generated gold standard references for a set of 330 (sentence, table) pairs gathered from a variety of source documents written for general audiences (e.g. Pew research reports [4], articles from the Economist magazine [2]) and for computer science researchers (e.g. ACL papers). Our pipeline correctly extracts 48.8% of the references. An additional 30.5% contain only false negative errors – the reference is incomplete and missing one or more table cells, while the remaining 20.7% contain false positive errors – the reference includes extraneous table cells and could therefore mislead readers.

We also conduct a controlled user study comparing our interactive document reader with a baseline reader that does not link text with tables. We find that despite the errors in reference extraction, when using our interface, participants match sentence text to table cells 26.4% more accurately and spend 22.9% less time than when using the baseline interface. These results suggest that automatically extracting and highlighting the links between document text and table cells reduces the split attention problem and facilitates reading the whole document. Our study also finds an asymmetry in the effects of FP errors and FN errors on this matching task. Participants are 23.2% less accurate and 27.8% slower at matching sentence text to table cells when the highlighted reference contains an FP error compared to highlighting a reference that contains only FN errors. This asymmetry suggests that FP errors are far more harmful to readers than FN errors because FP errors are misleading, while FN errors only omit information.

RELATED WORK

Our work builds on two main areas of prior work; (1) techniques designed to facilitate document reading, and (2) table parsing for question answering.

Techniques for Facilitating Document Reading

While documents often present data in the form of tables and graphics, reading and understanding the data in the context of the related text is challenging [17, 24]. Recently, researchers have developed techniques to better support such reading tasks. For example, several systems attempt to extract data and visual encodings from charts to improve data analysis and chart reading experiences [33, 19, 22, 32, 20]. Kong and Agrawala [23] further propose graphical overlay techniques to improve readability of existing visualizations after extracting the underlying data and encodings. However, these works primarily focus on improving the readability of visualizations while we focus on improving the readability of text and tables.

The most closely related work to ours is from Kong et al. [24] who extract references between text and visualizations (they only demonstrate their approach on bar charts) using a crowdsourcing pipeline. Our work differs in two ways: (1) Instead of relying on crowd workers to annotate the references, we present an automated algorithm for extracting references between text and tables. (2) Instead of working only with bar charts representing numeric data, the tables we work with can contain data of a variety of different types including text, number, money, time, etc.

Table Parsing for Question Answering

Govindaraju et al. [17] observe that tables are often embedded in text documents, and that the surrounding text contains information that is not present in the table. They present a technique for extracting the information in the surrounding text and joining it with the information in the table to improve the table schema. While our work is based on a similar observation, instead of improving table schema, we focus on extracting references between the text and table cells.

Researchers have also developed techniques for parsing tables into semi-structured representations that support automatic retrieval of answers to simple queries [10, 31, 30, 11]. Pimlikar et al. [31] and Cafarellla et al. [10] take a list of keywords and a set of tables as input and produce a ranked list of tables where the columns are relevant to the given keywords. Chen et al. [11] extract the hierarchical structures from tables in a spreadsheet format. While our approach is related to this line of work, we focus on parsing table struture and natural language sentences in the body text of PDF documents to extract references between the sentences and table cells.

Pasupat and Liang [30] take a table and a question specified in natural language as input and answer it based on the information in the table. The challenge here is to semantically understand questions and convert them into a sequence of arithmetic and logical operations. Our reference extraction problem does not involve building such arithmetic and logical expressions. Instead, we link sentences in the text to table cells using syntactic and semantic matching.

AUTOMATIC REFERENCE EXTRACTION

Our automatic reference extraction algorithm takes a PDF document as input and outputs references between each sentence in the text and cells in a table (Figure 2). The algorithm first breaks the main body text in the PDF into sentences using

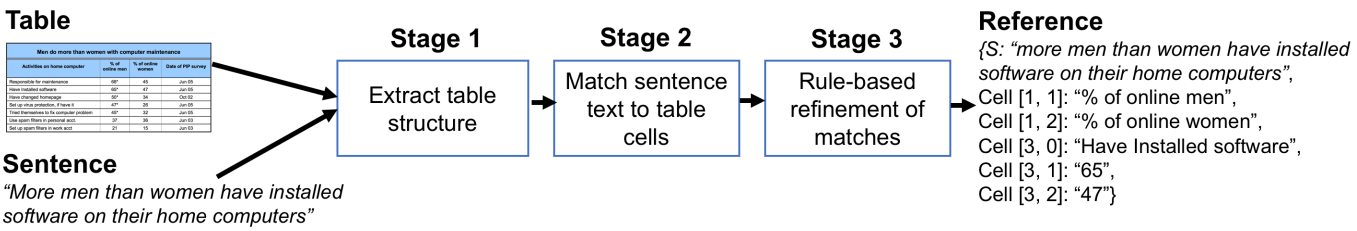
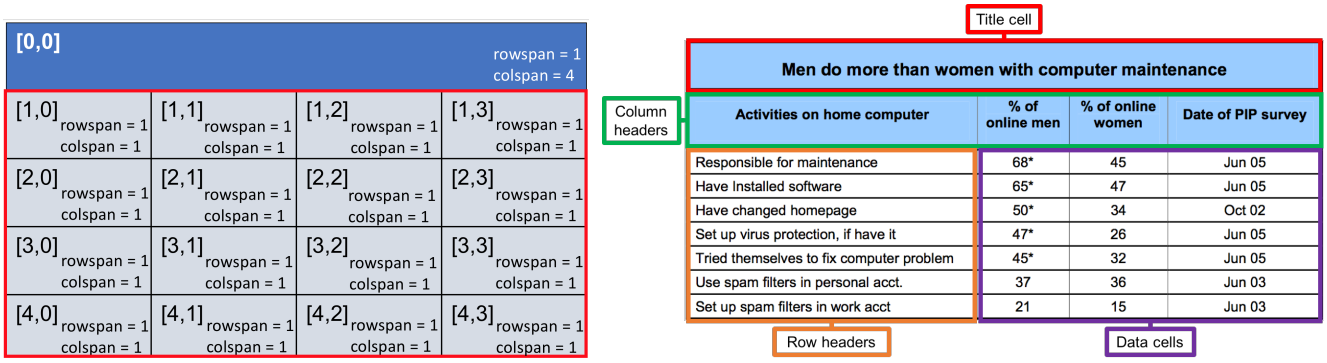


Figure 2. Our automatic reference extraction pipeline. The input to the pipeline is a (sentence, table) pair and the output is a reference matching the sentence with a corresponding set of table cells it refers to. The reference includes both the cell indices (Figure 3a) and cell contents.



(a) Row and column indices and `span` values (b) Cell types (title, headers, data)

Figure 3. In Stage 1 of our pipeline, we first compute the spatial row and column indices [rowindex, colindex] as well as the rowspan and colspan values for each cell (a). Specifically, we set [rowindex,colindex] = [0,0] for the top left cell and process the HTML table tags from top to bottom, left to right, incrementing colindex or rowindex each time we encounter a `</td>` or `</tr>` tag respectively. We similarly annotate each cell with the HTML `<rowspan>` and `<colspan>` information. Later in Stage 1, we use this spatial information to identify the regular subgrid (red outline). We then classify each cell of the table as a title cell, header cell or data cell using the span and subgrid structure (b).

the Stanford CoreNLP toolkit [27]. It also identifies and extracts all the tables in HTML format using Adobe Acrobat Reader [1]. Our algorithm then takes each (sentence, table) pair as input and applies a three stage pipeline to output either the set of cells the sentence refers to, or a null reference if there is no correspondence between the sentence and table.

Stage 1: Extract Table Structure

The first stage of our pipeline analyzes the input table to extract low-level information about the (1) spatial indices and spans, (2) data type (e.g. text, number, percentage, etc.) and (3) cell type (title, header, data) of each table cell. It also (4) normalizes the values held in each cell to a standardized format. Stages 2 and 3 of our pipeline use this low-level information to build the reference between the sentence and the table.

Compute spatial indices and spans of each cell

Given an input HTML table, we analyze the `<tr>` tags corresponding to each table row and `<td>` tags corresponding to each cell to generate row and column indices [rowindex, colindex] as well as the rowspan and colspan for each table cell (Figure 3a). The resulting indices encode the spatial position of the cells relative to one another and the spans indicate cells that span more than one row or column.

Identify data type of each cell

We classify the data type of each cell into one of six categories:

- ***Money:*** numeric value that represents an amount of money in some currency (e.g. "$10").
- ***Percent:*** numeric value that represents a percentage (e.g. "10%", "3 percent").
- ***Date:*** time value at granularity greater than one day (e.g. "1980/01/01", "April 2014").
- ***Time:*** time value at granularity finer than a day (e.g. "11:12", "11 o'clock", "15 sec").
- ***Number:*** numeric value that does not represent money, percent, date or time (e.g. a count, a rank).
- ***Text:*** text that does not fall into any other category.

To obtain this data type information, we apply the 7-class model of the Stanford Named Entity Recognizer [14] which labels each cell as either a Location, Organization, Person, Money, Percent, Date or Time. We ignore the Location, Organization and Person labels as the later stages of our pipeline do not need this information. We label any cells that do not fall into the Money, Percent, Date or Time categories as either Text or a Number depending on whether the cell contains a numeric value or it includes additional text.

Identify cell type of each cell

Tables commonly contain three types of cells (Figure 3b):

- ***Title cells*** are sometimes included as a part of a table and describe its overall contents.
- ***Header cells*** often appear at the top of columns (or left side of rows) and provide metadata describing the cells in the column (or row).
- ***Data cells*** appear in all tables as they hold the specific data values reported in the table.

Classifying table headers and titles versus data cells is challenging as their formats can vary from document to document or even table to table within a document [13]. In fact, we analyzed a collection of example tables from a variety of PDF

documents (newspaper articles, research papers, reports, etc.) and found that they use a variety of formats to distinguish titles and headers from data cells. In general, however, we found that for most tables the titles and headers appear in the topmost rows and/or the leftmost columns of the table and can sometime span multiple rows or columns. In contrast, data cells usually appear in the lower right part of the table and form a *regular grid* at the finest level of granularity (i.e. the cells do not span multiple rows or columns).

Based on these observations, we classify the cell type of each cell in the table in a two step process. First, we label any irregular rows or columns in the table – i.e. rows or columns that contain cells spanning more than one column or row, respectively (e.g. topmost row of Figure 3a). The remaining unmarked cells then form a regular grid (e.g. subgrid outlined in red in Figure 3a). We assume that the topmost row and leftmost column of this regular grid are headers at the finest level of granularity, and label all other cells within the regular grid as data cells. If the topmost header row contains a single cell that spans all the columns, we label it a title cell (Figure 3b).

Some tables do not contain row or column headers at the finest level of granularity. To properly handle such tables, we further rely on the assumption that the data type of header cells is often different from the data type of data cells. In many tables for example, header cells contain text while the data cells contain numbers. Therefore, we check if the cells in the topmost row and the leftmost column of the regular grid contain the same data type as the cells immediately below or to the right, respectively. If the data type is the same, we re-classify the finest granularity header cells as data cells.

Document authors sometimes leave data type information out of the data cells and only include it in the corresponding header cell. For example, the column header "% of online men" (Figure 3b) suggests that the data cells in the column are percentages, but the data cells only contain numeric values. To identify the data type of these cells, we parse header cells using a variety of common regular expressions (e.g. '% of', 'in $', 'in USD', etc.) and then propagate the data type information to the data cells in the corresponding columns or rows.

Normalize cell values

Authors sometimes put the order of magnitude of data values (e.g. billions, millions, etc.) into a header so that the table remains concise (Figure 4). To identify such order of magnitude information we again parse the header cells using common order of magnitude expressions (e.g. 'billions', '(B)', 'mill', etc.) and propagate the information to the corresponding data cells.

Largest online sales categories	*2002 holiday season sales in millions*	*% change vs. the same period a year ago*
Computer Hardware	$1,630	-1%
Apparel & Accessories	$1,455	31%
Consumer Electronics	$1,027	21%
Office	$576	-18%
Home & Garden	$555	78%
Books	$397	8%
Toys	$396	61%
Event Tickets	$250	16%
Sports & Fitness	$233	54%
Jewelry & Watches	$216	45%

Figure 4. This table includes an order of magnitude term 'million' in a column header. In the normalization step, we propagate this magnitude to the data cells in the column.

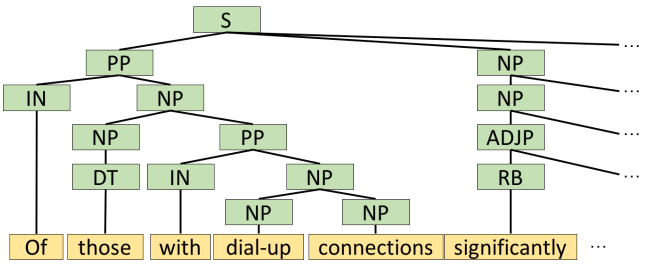


Figure 5. The phrase tree generated by the Stanford CoreNLP constituency parser for the sentence *"Of those with dial-up connections, significantly more men than women said they were interested in high-speed connections"*. Each subtree indicates a phrase and our syntactic matching algorithm finds a match between the phrase 'dial-up connections' and a cell in the table of Figure 1(right).

Stage 2: Match Sentence Text to Table Cells

In the second stage, we match the text of the input sentence to a corresponding set of cells in the input table using a combination of four different strategies using natural language processing (NLP) techniques. The first three strategies are designed to find matches against cells that contain text, while the fourth strategy is designed to find matches against cells that contain the other numeric data types.

Matching text cells based on unique words

Document authors often reference a specific table cell by including words in the sentence that uniquely appear in that cell and no other cell in the table. Consider the sentence *"However, mirroring the overall softness of the tech sector, sales of computer hardware decreased 1% versus a year-ago to $1.6 billion."* and the table in Figure 4. The terms 'computer' and 'hardware' appear in only one cell and it is likely that the sentence refers to it. We algorithmically implement this matching strategy by removing stop words from both the sentence and the table and lemmatizing the remaining words. Then for each cell, we store only the unique words, which do not appear in any other table cell. Finally, we match the remaining sentence words with the unique words in each table cell to identify a set of cells that the sentence is likely to be referencing.

Unfortunately, this simple matching strategy can produce incorrect references. Consider the sentence *"Of those who said they had virus protection on their home computers, significantly more men than women said they were responsible for setting up the protection"* and the table in Figure 3b. The word 'responsible' matches with the cell containing "Responsible for Maintenance", while the words 'virus' and 'protection' match with the cell containing "Set up virus protection, if have it". Only the second match is relevant to the sentence. We use syntactic and semantic analyses to better handle such cases.

Matching based on syntactic analysis

Syntactic analysis identifies the hierarchical phrase structure of a sentence as well as the grammatical dependencies between words. We use this syntactic structure to improve our matching algorithm. We first apply the constituency parser in the Stanford CoreNLP toolkit [27] to the input sentence to obtain its phrase tree (Figure 5). We then traverse the phrase tree breadth-first, starting at the root, and check if the entire

Reasons	% of online men	% of online women
Membership news and info	75	77
Discuss issues	72*	65
Group activity participation	71	71
Nurture member relationships	46	52

Figure 6. The sentence *"Significantly more men than women think that talking about topics is an important reason to email with these special interest groups."* matches the cell text "Discuss issues", because the sentence phrase 'talking about topics' has the same meaning as the cell text even though they have no words in common. Our semantic similarity matching strategy detects this match.

.

phrase in the current subtree (after removing stop words and lemmatization) is uniquely contained within a single cell of the table – every word in the phrase must appear in exactly one cell. If such a unique cell exists, we add it to the reference.

Using this approach, we can identify references where individual sentence words do not uniquely match to a single table cell, but multi-word phrases do uniquely match. Consider the sentence *"Of those with dial-up connections, significantly more men than women said they were interested in high-speed connections"* with respect to the table in Figure 1(right). The words 'dial-up' and 'connections' appear separately in multiple table cells, but both words in the phrase 'dial-up connections' appear together in only one cell. Our syntactic analysis strategy identifies the matching cell correctly.

Matching based on semantic analysis

Sometimes, a sentence phrase and the words in a cell have the same meaning, but do not have any words in common. Consider the sentence *"Significantly more men than women think that talking about topics is an important reason to email with these special interest groups"* and the table in Figure 6. The sentence phrase 'talking about topics' has the same meaning as the cell containing "Discuss issues", yet none of the words match and neither of our previous strategies would match them. To better handle such cases, we analyze the semantic similarity (i.e. similarity in meaning) between sentence phrases and the words in each cell.

We modify our syntactic matching strategy to compare each sub-phrase in the breadth-first traversal of the phrase tree using a distance based on *word2vec* – a vector model of words that encodes semantics. We use the pre-trained model of Mikolov et al. [28] which was trained on parts of the Google News dataset [6] and produces vectors with 300 dimensions.

Specifically, we look up the word2vec vector for each word in the sentence phrase (after stop word removal and lemmatization) and sum them to generate a vector v_s representing the phrase. We similarly look up and sum the vectors for each word in the cell text to generate v_c, and then compute the cosine similarity between these vectors as

$$cos(\boldsymbol{v}_s, \boldsymbol{v}_c) = \frac{\boldsymbol{v}_s \cdot \boldsymbol{v}_c}{||\boldsymbol{v}_s|| \cdot ||\boldsymbol{v}_c||} \quad (1)$$

The word2vec model is designed so that the closer this cosine similarity is to 1, the greater the semantic similarity between the sentence phrase and the cell text. Therefore, whenever the cosine similarity between them is greater than a threshold τ (set empirically to 0.75), we treat them as a semantic match. This procedure correctly handles the example in Figure 6 because the semantic similarity between the sentence phrase 'talking about topics' and the cell text "Discuss issues" is above our semantic matching threshold τ.

Handling cells containing numeric and time values

To match sentence text to cells containing numeric values (i.e. numbers, percents or money), we first detect all strings in the sentence that represent numbers using the Stanford Named Entity Recognizer [14] and convert them into numerals (e.g. 'five million' is converted to 5,000,000).

Document authors often refer to table cells containing numeric values by rounding their rightmost significant digit. For instance, the sentence phrase *"about 1.5 meters"* may be used to refer to a table cell containing the value 1.53 meters. In some cases, the sentence phrase may also suggest the direction of rounding – either up or down. For example, the phrase *"more than 5 million"* may be used to refer a table cell containing the value 5,700,000. Thus, whenever we encounter a numeric value in the sentence text, we examine the surrounding words to check whether they indicate a rounding direction (up, down or nearest) – e.g. 'more than' indicates the value in the sentence has been rounded down. Then, if we do not find an exact match to the numeric value in the sentence, we compare the rounded value. As shown in Figure 4, this approach allows us to match the dollar amount in the sentence *"... sales of computer hardware decreased 1% versus a year-ago to $1.6 billion"* to the topmost data cell in the second column containing "$1,630".

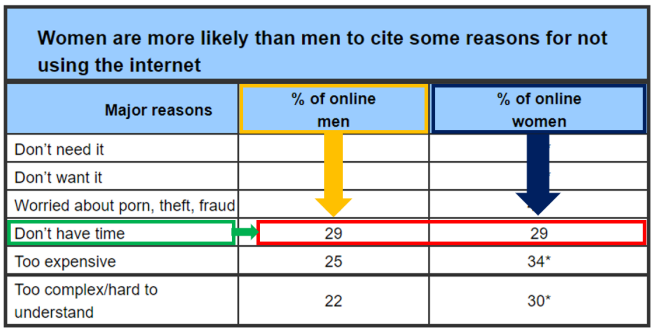
We have found that document authors use a variety of formats to express dates (e.g. '01/01/1980' and 'Jan 01, 1980'), time (e.g. '14:02' and '2:02 PM'), and proportions (e.g. '20%' and '1 in 5'). To handle such variability, we detect dates, time and proportions within the sentence text using regular expression templates (e.g. 'dd/mm/yy', 'dd-mm-yyyy', 'hh:mm:ss', etc.) and normalize them to a standardized format so that our algorithm can correctly match equivalent expressions.

Stage 3: Rule-based Refinement of Matches

While many of the matches produced in Stage 2 are correct, because Stage 2 does not consider table structure (i.e. cell type – title, header, data of each cell), it can miss matches between the sentence and cells and it can incorrectly match the sentence to irrelevant cells. For instance, implicit references as in Figures 1(right) and 7, occur when a sentence only describes the row and column header cells in the text, leaving it up to the reader to identify the data cells that fall in the intersection of the these rows and columns. Our matching algorithm in Stage 2 would miss the matches to these data cells. The third stage of our pipeline is designed to handle such implicit references and also remove irrelevant matches based on table structure.

Rule 1: Add implicit data cells in the intersection of headers

To properly handle implicit references, our first rule considers all of the row and column headers returned by the matching algorithm in Stage 2 and automatically adds the data cells that fall in the intersection of the corresponding rows and columns



Women are more likely than men to cite some reasons for not using the internet		
Major reasons	% of online men	% of online women
Don't need it		
Don't want it		
Worried about porn, theft, fraud		
Don't have time	29	29
Too expensive	25	34*
Too complex/hard to understand	22	30*

Figure 7. The sentence *"Equal numbers of men and women said they didn't have time"* implicitly refers to the data cells containing the value 29. However, the matching stage of our pipeline (Stage 2) only matches the row and column headers (green, yellow, blue outlines) to the sentence text. In Stage 3, we apply the *add implicit data cells rule* to correctly add in the implicit data cells (red outlines) to the reference.

Men push the tech edge more than women.			
Entertainment activities	% of online men	% of online women	Date of PIP survey
Download computer programs	48*	31	Jun 05
Text messaging	33	37	Sep 05
Wireless log on	27	22	Nov 04
Share files	25	28	Jun 05
Remix files	21*	15	Feb 05
Use webcam	19*	13	Mar 05
Maintain website	16*	11	May 03
Own iPods or Mp3 players	13*	9	Mar 05
Made VOIP call	13*	9	Feb 04
Create a blog	11	8	Sep 05

Figure 8. The sentence *"In March 2005, 13% of men owned iPods or Mp3 players"*, matches with all the cells outlined in red and green after the matching stage (Stage 2) of our pipeline. However, only the cells with green outline are correct matches. In Stage 3, we remove the cells with red outline based on the rule that cells which do not appear in the intersection of row and column headers should be removed.

to the set of matched cells. Applying this rule on the example in Figure 7 correctly adds the implicitly referenced data cells.

Rule 2: Remove data cells not in the intersection of headers

In some tables, the same value may appear in multiple data cells and if a sentence contains the value, our matching algorithm (Stage 2) identifies all such cells as a match to the sentence even though some of them may be irrelevant (Figure 8). But if the sentence also refers to the row and column headers, we can use the table structure to remove the irrelevant data cell matches. Our second rule only retains data cells that lie at the intersection of matched row and columns headers and eliminates all other data cells.

Rule 3: Add implicit header cells if data cells match uniquely

Header cells are sometimes referenced implicitly as well. Consider the sentence *"34% of women cited cost as the reason for not using the internet"* and the table in Figure 7. Our matching stage (Stage 2) matches the sentence text 'women' with the header cell "% of online women" and the sentence text '34%' with a unique data cell "34" But the sentence does not explicitly reference the row header cell "Too expensive" and our semantic matching strategy does not find a matching sentence phrase that is above its match threshold. We handle such implicit references to header cells by automatically adding the header cells whenever a single data cell matched uniquely within the table in Stage 2. In this case, since the single data cell "34" is uniquely matched, this rule allows us to correctly include the row header cell "Too expensive."

Rule 4: Remove potentially irrelevant header cells

In some cases, our matching algorithm in Stage 2 finds matches between the sentence text and row and column header cells, but the sentence also contains numeric data values that do not appear in the table. Consider the example sentence *"58% of men and 47% of women said they know how to upload images or other files to a website so others could see them"* with respect to the table in Figure 8. In Stage 2, we obtain matches to the columns "% of online men" and "% of online women". However, the numeric data values given in the sentence 58% and 47% do not appear in any of the table cells. In such cases, our fourth rule removes the header cells based on the assumption that the sentence is unlikely to be related to the table. In this case, the rule removes "% of online men" and "% of online women" from the reference.

PIPELINE EVALUATION

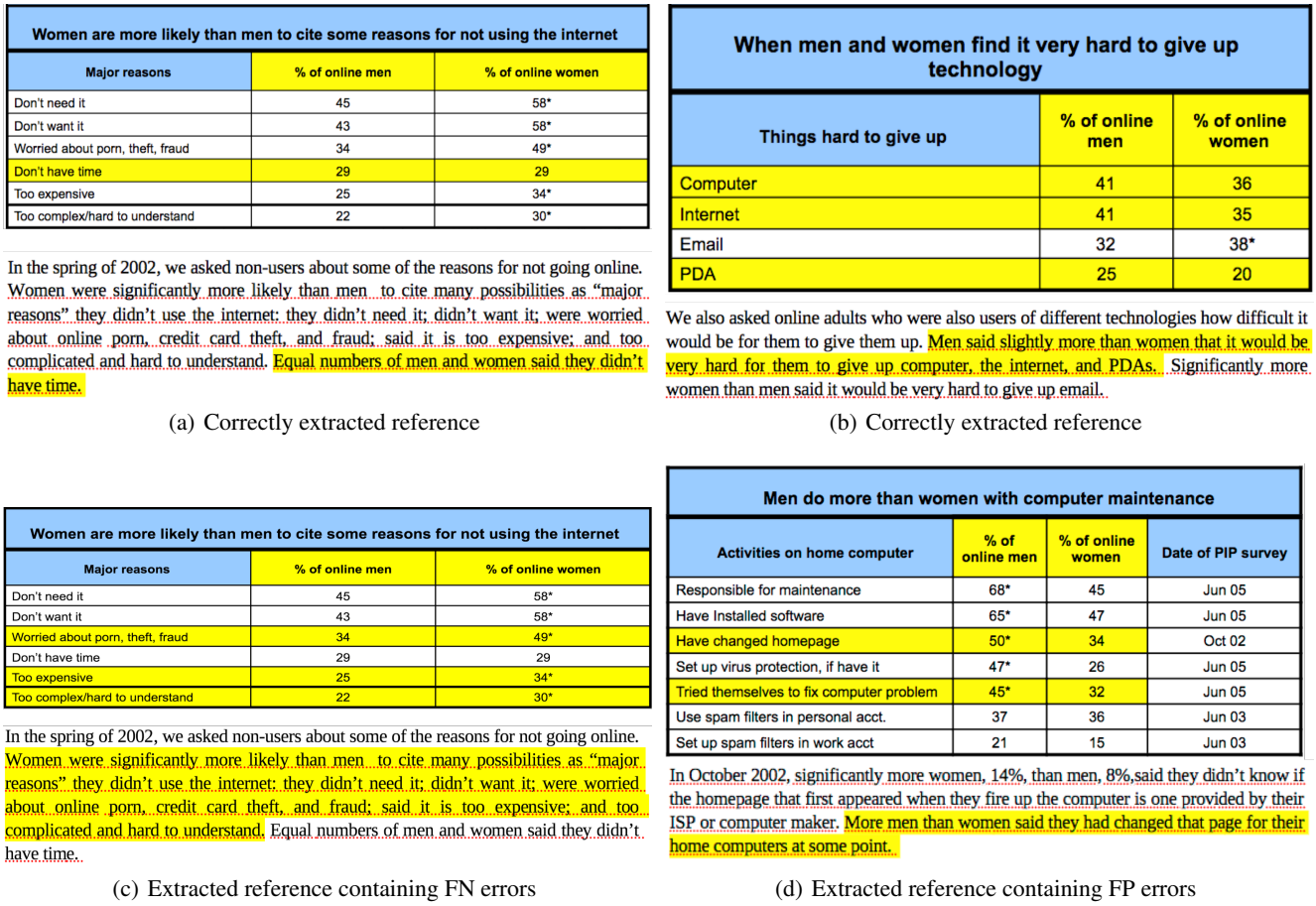
Figures 1, 9 and 12 show references generated using our automatic reference extraction pipeline. To quantitatively evaluate the accuracy of our automatic reference pipeline, we gathered a representative sample of (sentence, table) pairs from documents written for general audiences as well as scientific papers written for researchers. We obtained a gold reference set for each pair and then compared the results from our reference extraction pipeline to the gold reference set.

Corpus

To build the representative sample of (sentence, table) pairs, we gathered two sets of PDF documents written for different audiences. Our *Pew* dataset contains 10 research reports written for general audiences and published by Pew Research [4] in the area of public policy. Our *Academic* dataset contains 6 research papers written for computer science researchers from the ACL conference [18, 34, 16, 25, 12, 26]. Since most sentences in a document are unrelated to any table within it, we manually identified tables as well as paragraphs related to these tables from the corpus. Thus, we could ensure that many of the sentences would reference the tables, but since we took entire paragraphs, we could also be sure that some sentences would not reference the tables. Table 1 summarizes the number of tables, paragraphs and (sentence, table) pairs we extracted for each dataset.

Dataset	# Docs	# Tables	# Paras	# (sentence, table) pairs
Pew	10	26	35	127
Academic	6	11	14	72
Kong [24]	18	35	49	139

Table 1. Summary of the three datasets we use to evaluate our pipeline. The *Pew* and *Kong* datasets are culled from documents written for general audiences while the *Academic* dataset is from computer science research papers.



Women are more likely than men to cite some reasons for not using the internet		
Major reasons	% of online men	% of online women
Don't need it	45	58*
Don't want it	43	58*
Worried about porn, theft, fraud	34	49*
Don't have time	29	29
Too expensive	25	34*
Too complex/hard to understand	22	30*

In the spring of 2002, we asked non-users about some of the reasons for not going online. Women were significantly more likely than men to cite many possibilities as "major reasons" they didn't use the internet: they didn't need it; didn't want it; were worried about online porn, credit card theft, and fraud; said it is too expensive; and too complicated and hard to understand. Equal numbers of men and women said they didn't have time.

(a) Correctly extracted reference

When men and women find it very hard to give up technology		
Things hard to give up	% of online men	% of online women
Computer	41	36
Internet	41	35
Email	32	38*
PDA	25	20

We also asked online adults who were also users of different technologies how difficult it would be for them to give them up. Men said slightly more than women that it would be very hard for them to give up computer, the internet, and PDAs. Significantly more women than men said it would be very hard to give up email.

(b) Correctly extracted reference

Women are more likely than men to cite some reasons for not using the internet		
Major reasons	% of online men	% of online women
Don't need it	45	58*
Don't want it	43	58*
Worried about porn, theft, fraud	34	49*
Don't have time	29	29
Too expensive	25	34*
Too complex/hard to understand	22	30*

In the spring of 2002, we asked non-users about some of the reasons for not going online. Women were significantly more likely than men to cite many possibilities as "major reasons" they didn't use the internet: they didn't need it; didn't want it; were worried about online porn, credit card theft, and fraud; said it is too expensive; and too complicated and hard to understand. Equal numbers of men and women said they didn't have time.

(c) Extracted reference containing FN errors

Men do more than women with computer maintenance			
Activities on home computer	% of online men	% of online women	Date of PIP survey
Responsible for maintenance	68*	45	Jun 05
Have Installed software	65*	47	Jun 05
Have changed homepage	50*	34	Oct 02
Set up virus protection, if have it	47*	26	Jun 05
Tried themselves to fix computer problem	45*	32	Jun 05
Use spam filters in personal acct.	37	36	Jun 03
Set up spam filters in work acct	21	15	Jun 03

In October 2002, significantly more women, 14%, than men, 8%,said they didn't know if the homepage that first appeared when they fire up the computer is one provided by their ISP or computer maker. More men than women said they had changed that page for their home computers at some point.

(d) Extracted reference containing FP errors

Figure 9. References extracted by our automatic reference extraction pipeline. (a) Correctly extracted reference for the sentence "*Equal numbers of men and women said they didn't have time*." (b) Correctly extracted reference for the sentence "*Men said slightly more than women that it would be very hard for them to give up computer, the internet, and PDAs*." (c) Reference containing false negative errors (missing cells) for the sentence "*Women were significantly more likely than men to cite many possibilities as "major reasons" they didn't use the internet: they didn't need it; didn't want it; were worried about online porn, credit card theft, and fraud; said it is too expensive; and too complicated and hard to understand.*" Because the pipeline removes the stop words 'do', 'not', 'need', 'want', and 'it', it misses the header rows "Don't need it" and "Don't want it." (d) Reference containing false positive errors (includes irrelevant cells) for the sentence "*More men than women said they had changed that page for their home computers at some point.*" Our pipeline detects an extra row because the word 'computer' appears in the sentence and in a single cell "Tried themselves to fix computer problem."

For comparison, we include a third dataset from Kong et al. [24] that contains (sentence, table) pairs from 18 general audience documents including news sources like the Economist [2] and the Guardian [3]. Together, the documents in our datasets cover a range of writing styles and table usages.

Gold Reference Set

We used an iterative process to create a gold reference set for the (sentence, table) pairs in the *Pew* and *Academic* datasets. First, two authors from our research team independently identified references between the (sentence, table) pairs following the reference annotation guidelines of Kong et al. [24]. They then resolved each inconsistency by explaining their logic in producing the reference. They then worked together to develop a consensus reference. Finally, a third author scrutinized the resulting references and initiated a second round of debate for each reference that he disagreed with. After a thorough discussion between all three authors, they reached a final consensus about the set of cells to include as the gold reference set for each sentence. For the *Kong* dataset, we used the gold references provided by Kong et al. [24].

Pipeline Performance and Accuracy

Across all three datasets, our reference extraction pipeline took an average of 258.38 ms to process each (sentence, table) pair on a 2.5Ghz MacBook Pro with an Intel Core i7 processor and 16GB RAM. Stage 1 took an average of 233.89 ms per table, Stage 2 took 208.14 ms per sentence, and Stage 3 took 0.42 ms per sentence.

To compute the accuracy of our pipeline, we compare the results it generates to the gold references. Specifically, for each sentence, we compare our automatically generated reference A to the corresponding gold reference G and categorize the results as follows:

- **Correct reference:** our pipeline generates the exact same set of table cells as in the gold reference, i.e. $G = A$.
- **False negative (FN):** our pipeline generates a reference that is missing some cells that are included in the gold reference, i.e. $A \subset G$.

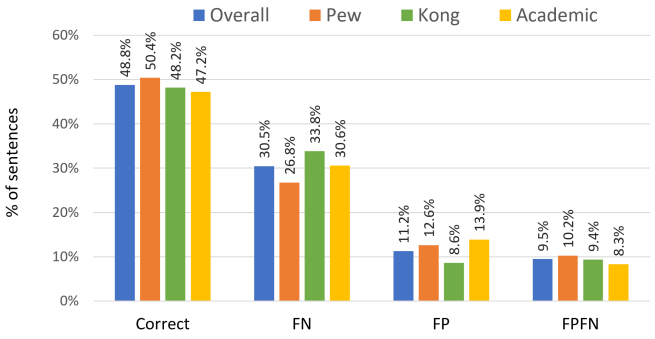


Figure 10. Comparison of correct, FN, FP and FPFN references produced by our complete pipeline for each of the datasets, Pew (orange), Kong (green) and Academic (yellow) as well as the overall combination of all three datasets (blue).

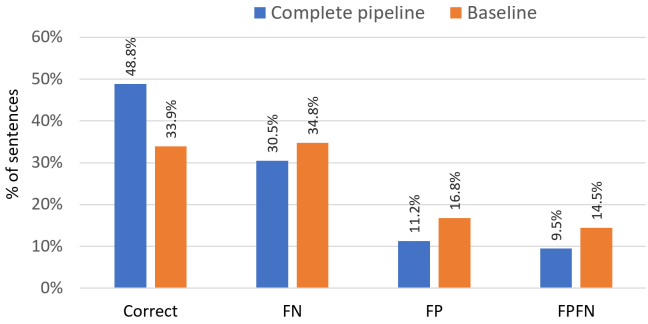


Figure 11. Comparison of correct, FN, FP and FPFN references produced by our complete pipeline and a baseline method using only the unique words matching strategy combining all three datasets.

- **False positive (FP):** our pipeline generates a reference that includes extraneous cells that are not in the gold reference, i.e. $G \subset A$.
- **False positive + False negative (FPFN):** our pipeline generates references with both false negatives and false positives, i.e. $G \not\subset A$ and $A \not\subset G$.

As shown in Figure 10, we find that overall (blue bars) across all three datasets, our complete pipeline generates 48.8% correct references, 30.5% references that contain only FN errors, 11.2% references that contain only FP errors, and 9.5% references that contain both FN and FP errors. Moreover, the accuracy numbers are similar across the three datasets despite the fact that they contain different kinds of writing meant for different audiences. This result suggests that the performance of our pipeline is somewhat independent of writing style.

For the *Kong* dataset, we also compare our pipeline with the crowdsourcing pipeline of Kong et al. [24] that combines references generated by multiple workers into a single set, using clustering and merging techniques. Their approach produces 71.2% correct references, 8.6% FN errors and 18.8% FP errors and 1.4% FPFN errors. While their crowdsourcing pipeline produces 22.4% more correct references than our automatic pipeline, their increase in accuracy comes at the cost of significantly more annotation effort as they require multiple crowd workers to independently generate references for each (sentence, table) pair.

Figure 11 compares the accuracy of our reference extraction pipeline to a baseline version of our pipeline that only includes the matching on unique words strategy and does not include other strategies in Stage 2 or the rule-based refinements of Stage 3. This comparison shows that the complete pipeline with the syntactic and semantic matching, as well as the rule-based refinement, provides a substantial improvement in the percentage of correct references over the baseline.

INTERACTIVE DOCUMENT READER

The goal of our interactive document reader (Figure 12) is to assist viewers by displaying references between the document text and the tables as they read the document. Given a PDF document with a set of such references, our reader underlines each sentence that references a table in red. Clicking on such a sentence highlights it and the table cells it refers to in yellow. Similarly, clicking on a cell highlights all the sentences that refer to it. Clicking anywhere else on the document removes the highlight.

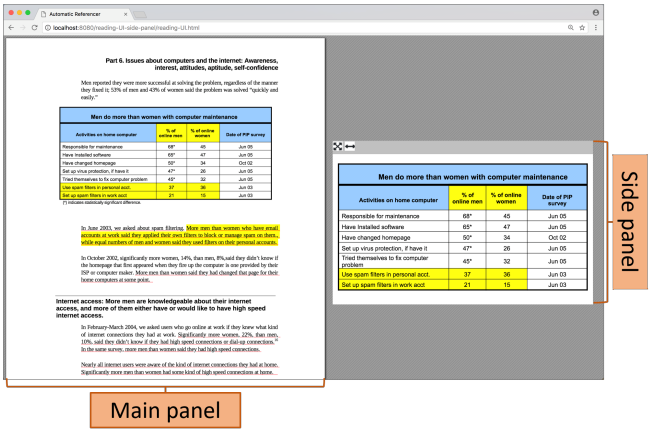


Figure 12. Our interactive document reader contains a main panel showing the document and a side panel showing the table most relevant to the sentences at center of the main panel. Red underlines indicate sentences that refer to cells in a table. Clicking on such a sentence highlights it and the table cells it refers to in yellow. Clicking on a cell highlights all sentences that refer to it. Clicking anywhere else on the document removes the highlight.

To reduce the problem of split attention that occurs when a table is located relatively far away in the document from the referencing text, we include a side panel that replicates the table most relevant to the sentence at the center of the main panel. As the user scrolls through the pages in the main panel, the most relevant table in the side panel automatically updates. The table is scaled by default to fit in the panel, but clicking the expand button expands the table to full size. The table is a fully interactive copy of the table in the main panel and clicking on a cell in either table highlights the relevant sentences in the document and vice versa.

USER STUDY

We conducted a user study to compare our interactive document reader with automatic linking of sentences to table cells to a baseline reader (similar to Adobe Reader) that does not provide such links. We consider two main hypotheses:

H1: Despite the errors produced in our automatic reference extraction pipeline, our interactive document reader will help users locate table cells relevant to sentences in the text more accurately and quickly than the baseline reader.

H2: Since false positive (FP) errors can mislead readers by connecting sentences to incorrect table cells, they will cause more harm (lower speed and accuracy) than false negative (FN) errors which simply force readers to manually identify the connection between sentences and table cells.

Study Design

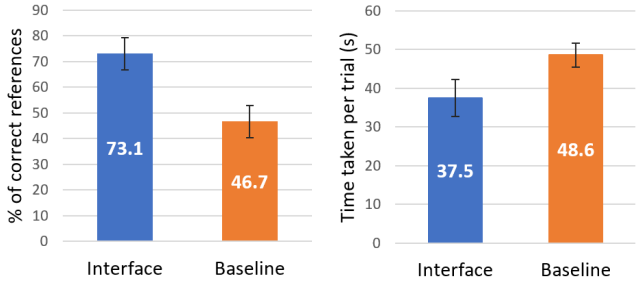
We used a within-subjects study design. We sampled two groups of 12 (sentence, table) pairs from our corpus such that the distribution of error types (correct, FN, FP, FPFN) in each group roughly matched the overall distribution produced by our automatic reference extraction pipeline (6 correct, 4 FN, 1 FP, 1 FPFN). We used the first group of references to generate 12 *interface condition* trials and the second group to generate 12 *baseline condition* trials. Each trial presented a single (sentence, table) reference pair, where the sentence was underlined in red and the paragraph containing the sentence was shown for context. The interface condition included the interactive reference highlighting of our interactive document reader, while the baseline condition did not include such highlighting. On each trial, the participant had to select the table cells referenced by the underlined sentence.

Study Procedure

We recruited 14 adult participants, all fluent in English, from three academic institutions. Each participant completed 24 trials, 12 in each condition. We counterbalanced the ordering of the conditions and randomized the ordering of trials within each condition for each participant to reduce ordering effects. Before running the experiment, participants went through a training session where they learned how to use both conditions and correctly complete the trial task. During the experiment, we measured the accuracy and speed of each trial. The participants were aware that we were measuring both time and accuracy, but we did not specifically ask them to prioritize speed or accuracy. After completing all 24 trials, we asked the participants to rate the helpfulness of our interface on a 5 point Likert scale and to express their opinions about the usefulness of the interface in a free-form text response. The experiment took about 45 minutes to complete and each participant received a $20.00 Amazon gift card for participating in the study.

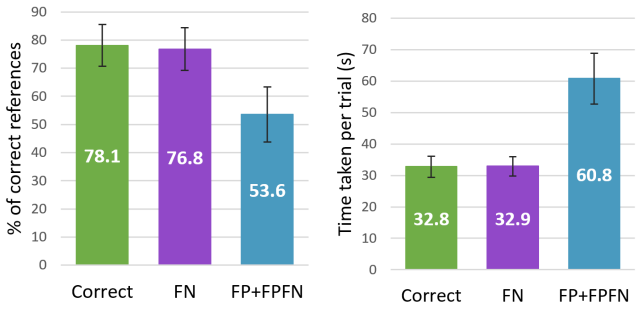
Results

We find that our interactive reading interface significantly outperforms the baseline interface in terms of accuracy and speed (Figure 13). Accuracy in the interface condition ($\mu = 73.1\%$, $\sigma = 23.61$) was 26.4% higher than in the baseline condition ($\mu = 46.7\%$, $\sigma = 19.67$, $t(13) = 7.57$, $p < 0.001$). On average it took participants 22.9% (11.13 seconds) less time in the interface condition ($\mu = 37.5$s, $\sigma = 11.69$) than in the baseline condition ($\mu = 48.6$s, $\sigma = 17.96$, $t(13) = 4.12$, $p < 0.05$). In their subjective assessments of helpfulness of our interface for reading documents (on a 5 point scale with 5 = very helpful), participants were generally positive ($\mu = 4.1$, $\sigma = 0.17$). In the free-form response, one of the participants who had the interface condition first commented that after the transition to the baseline condition, the increased amount of effort required to complete each trial was noticeable. Together, these results suggest that we can accept hypothesis H1.



(a) Accuracy (b) Time

Figure 13. Users are significantly more accurate and faster at matching sentence text to table cells using our interactive reference highlighting interface compared to a baseline interface. These results indicate that despite errors introduced by our reference extraction pipeline, our interface facilitates document reading overall.



(a) Accuracy (b) Time

Figure 14. Comparison of user accuracy in matching sentence text to table cells when our interactive interface presents references that are correct, that contain FN errors only and that contain at least one FP error. We find that FP errors are significantly more harmful (reduce accuracy and speed) than FN only errors and that the difference between presenting correct references and FN only is not significant.

We also find that presenting a reference containing FP errors in our interface harms accuracy and speed much more than presenting a reference containing only FN errors (Figure 14). Specifically, a one-way RM-ANOVA finds a significant effect for the types of references presented (Correct, FN – contains only FN errors, FP+FPFN – contains at least one FP error) on accuracy ($F(2,26) = 4.89$, $p < 0.05$). Participants suffered a 24.5% hit in accuracy when the presented reference contained an FP error ($\mu = 53.6\%$, $\sigma = 36.50$) compared to when the presented reference was correct ($\mu = 78.1\%$, $\sigma = 27.57$, $t(13) = 3.061$, $p < 0.005$). Similarly with FP errors they suffered a 23.2% reduction in accuracy compared to when the reference contained only FN errors ($\mu = 76.8\%$, $\sigma = 28.53$, $t(13) = 2.061$, $p < 0.05$). These results suggest that FP errors are far more harmful to accuracy than FN errors. In fact, we found that when comparing accuracy for references containing only FN errors to correct references, there is no significant difference.

We find similar results for speed. A one-way RM-ANOVA finds a significant difference in time required to complete the trial for the three types of references presented ($F(2,26) = 11.815$, $p < 0.001$). Participants took 28.0 seconds longer per trial when shown references containing FP errors ($\mu = 60.8$, $\sigma = 30.39$) than when shown correct references ($\mu = 32.8$, $\sigma = 12.59$, $t(13) = 3.372$, $p < 0.005$). Similarly, they took 27.8 seconds longer when shown references with FP errors than when shown references containing only FN errors ($\mu = 32.9$, $\sigma = 11.50$, $t(13) = 3.707$, $p < 0.005$). Moreover, we saw no significant drop in the speed between being shown correct references and being shown references with FN errors. Together these accuracy and speed results suggest that we can also accept hypothesis H2.

DISCUSSION

Our main goal in developing our interactive document reader was to reduce split attention when reading documents containing tables. Our user study finds that users can match document text to table cells more accurately and quickly using our interface than they can using a standard baseline document reader. This result indicates that our interface does reduce the split attention problem.

In addition to the controlled user study, we have observed a number of users as they interact with our document reader interface. They all agreed that our interface was easy to use and that they found the link connecting sentence text to table cells to be useful. One compared our interface to standard document viewers saying, "*The interface allows me to read the table while reading the text. Originally this was done in the text-then-table order, but this was parallelized, making it more efficient.*" Another said, "*In everyday life, if text includes tables, I would usually trust the text and not read the table too carefully, but this interface made me take time looking back and forth.*" These observations also suggest that our interface reduces split attention and facilitates document reading.

From the user study, we also found that presenting references containing FP errors is more harmful than presenting references containing only FN errors. We believe that this is because FP errors can actively mislead readers by matching text sentences to irrelevant table cells. In contrast, FN errors simply force readers to manually identify the connection between sentence text and table cells. In the free-form text response, one of the study participants wrote "*I would prefer to have missing information [than to have extra information] because I can always fill in the gaps.*" Being misinformed (FP errors) is much worse than being uninformed (FN errors).

Another implication of this finding is that while there is some room for improvement in the percentage of correct references produced by our pipeline, 48.8%, it may be best to focus future work on reducing the FP error rate of 20.7% before addressing the FN errors. Moreover, as our user study shows, extracting references between text and tables is challenging even for people. In the baseline condition, participants produced 46.7% correct references ($\sigma = 19.67$) on average, suggesting that our pipeline produces correct references at rates that are comparable to human performance.

LIMITATIONS AND FUTURE WORK

Although our automatic reference extraction pipeline provides good accuracy, there are several limitations that we would like to lift in future work.

First, while our work provides evidence that our pipeline generalizes across different document styles, a larger corpus containing additional types of documents (e.g. textbooks, news articles, etc.) would allow us to verify the generalizability of our pipeline across document types. Generating labeled data from the larger corpus would also pave the way for developing modern machine-learning based methods for this problem.

Second, since most sentences in a document are not related to any table, we manually identified paragraphs relevant to tables to ensure that there are a sufficient number of sentences referring to tables. Future work includes devising a paragraph-to-table matching method by using features such as mentions of the table number or the distance between the text and tables.

Third, our pipeline does not handle some complex references. While our work focuses on references within a single sentence, anaphora resolution techniques [29] and/or a document-level discourse parser [21] could be used to identify references that span multiple sentences. Some references also require external knowledge (e.g. table headers list the countries in the world and the text refers to 'Asian countries'). Knowledge bases such as DBPedia [7], Freebase [9] or Wolfram Alpha [5] could be used to resolve such references. References sometimes include clauses such as 'all but ...' (exclusion) or 'the second most common' (ranking) to indicate specific table cells. Applying compositional semantic parsing [30] may be one approach for resolving references that involve such logical operations.

Finally, additional user studies could help us further understand exactly why users were more accurate with our interface than without it. Such studies would need to be designed to elucidate the mechanisms by which our reference highlighting helps users. Moving beyond the table-cell selection task, future studies could also examine the user's ability to recall facts or search for information using our interface.

CONCLUSION

We have presented a fully automatic pipeline for extracting references between the text and tables in a document. Our pipeline includes three main stages that analyze the structure of the table, apply natural language processing techniques to match sentence text to table cells and refine the matches using the table structure. While our results are not perfectly accurate, the majority of errors are due to false negatives (missing cells), which we have found to be less harmful than false positives (misleading cells) in the user study. We believe that this work is an initial step towards more interactive documents that assist readers in absorbing their content by linking and presenting multiple sources of relevant information.

ACKNOWLEDGMENTS

The authors thank Sherry Ruan and Naoki Eto for early software developoment and Sean Liu for help with recording the videos. This work is supported by an Allen Distinguished Investigator award and by NSF award III-1714647. Dae Hyun Kim is partly supported by a Samsung Scholarship.

REFERENCES

1. 2018. Adobe Acrobat Reader. (2018). Retrieved April 02, 2018 from https://acrobat.adobe.com/us/en/acrobat/acrobat-pro.html.
2. 2018. Economist Graphic Detail. (2018). Retrieved April 02, 2018 from https://www.economist.com/blogs/graphicdetail.
3. 2018. Guardian DataBlog. (2018). Retrieved April 02, 2018 from https://www.theguardian.com/data.
4. 2018. Pew Research. (2018). Retrieved April 02, 2018 from www.pewresearch.org/.
5. 2018. Wolfram Alpha. (2018). Retrieved April 02, 2018 from https://www.wolframalpha.com.
6. 2018. Word embedding trained on Google News. (2018). Retrieved April 02, 2018 from https://code.google.com/archive/p/word2vec/.
7. Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A nucleus for a web of open data. In *Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference*. 722–735.
8. Paul Ayres and Gabriele Cierniak. 2012. *Split-attention effect*. Springer US, Boston, MA, 3172–3175.
9. Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, 1247–1250.
10. Michael J Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. 2008. WebTables: Exploring the power of tables on the web. *Proceedings of the VLDB Endowment* 1, 1 (2008), 538–549.
11. Zhe Chen, Michael Cafarella, Jun Chen, Daniel Prevo, and Junfeng Zhuang. 2013. Senbazuru: A prototype spreadsheet database management system. *Proceedings of the VLDB Endowment* 6, 12 (2013), 1202–1205.
12. Jianpeng Cheng, Siva Reddy, Vijay Saraswat, and Mirella Lapata. 2017. Learning structured natural language representations for semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 44–55.
13. Jing Fang, Prasenjit Mitra, Zhi Tang, and C Lee Giles. 2012. Table header detection and classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 599–605.
14. Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, 363–370.
15. Mareike Florax and Rolf Ploetzner. 2010. What contributes to the split-attention effect? The role of text segmentation, picture labelling, and spatial proximity. *Learning and Instruction* 20, 3 (2010), 216–224.
16. Jonas Gehring, Michael Auli, David Grangier, and Yann Dauphin. 2017. A convolutional encoder model for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 123–135.
17. Vidhya Govindaraju, Ce Zhang, and Christopher Ré. 2013. Understanding tables in context using Standard NLP toolkits. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. 658–664.
18. E Dario Gutierrez, Ekaterina Shutova, Tyler Marghetis, and Benjamin Bergen. 2016. Literal and metaphorical senses in compositional distributional semantic models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 183–193.
19. Jonathan Harper and Maneesh Agrawala. 2014. Deconstructing and restyling D3 visualizations. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. ACM, New York, NY, USA, 253–262.
20. Jonathan Harper and Maneesh Agrawala. 2018. Converting basic D3 charts into reusable style templates. *IEEE Transactions on Visualization and Computer Graphics* 24, 3 (March 2018), 1274–1286.
21. Shafiq Joty, Giuseppe Carenini, and Raymond T Ng. 2015. Codra: A novel discriminative framework for rhetorical analysis. *Computational Linguistics* (2015), 385–435.
22. Daekyoung Jung, Wonjae Kim, Hyunjoo Song, Jeong-in Hwang, Bongshin Lee, Bohyoung Kim, and Jinwook Seo. 2017. ChartSense: Interactive data extraction from chart images. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 6706–6717.
23. Nicholas Kong and Maneesh Agrawala. 2012. Graphical overlays: Using layered elements to aid chart reading. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (2012), 2631–2638.
24. Nicholas Kong, Marti A Hearst, and Maneesh Agrawala. 2014. Extracting references between text and charts via crowdsourcing. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, 31–40.
25. Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 146–157.

26. Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 1–10.
27. Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. 55–60.
28. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).
29. Ruslan Mitkov. 2014. *Anaphora resolution*. Routledge.
30. Panupong Pasupat and Percy Liang. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*. 1470–1480.
31. Rakesh Pimplikar and Sunita Sarawagi. 2012. Answering table queries on the web using column keywords. *Proceedings of the VLDB Endowment* 5, 10 (2012), 908–919.
32. Jorge Poco and Jeffrey Heer. 2017. Reverse-engineering visualizations: Recovering visual encodings from chart Images. In *Computer Graphics Forum*, Vol. 36. Wiley Online Library, 353–363.
33. Manolis Savva, Nicholas Kong, Arti Chhajta, Li Fei-Fei, Maneesh Agrawala, and Jeffrey Heer. 2011. ReVision: Automated classification, analysis and redesign of chart images. In *Proceedings of the 24th annual ACM symposium on User Interface Software and Technology*. ACM, 393–402.
34. Mingxuan Wang, Zhengdong Lu, Jie Zhou, and Qun Liu. 2017. Deep neural machine translation with linear associative unit. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 136–145.