Table Classification: an Application of Machine Learning to Web-hosted Financial Documents

Marc Vilain, John Gibson, Benjamin Wellner, and Rob Quimby

The MITRE Corporation
202 Burlington Rd.
Bedford, Mass. 01730 USA
{wellner, jgibson, mbv}@mitre.org , rob.quimby@students.olin.edu

Abstract

This paper presents learning-based techniques that support the processing of tables in HTML publications. We are concerned especially with classifying tables as to format and content, focusing on the domain of corporate financials. We present performance results based on multiple classification methods, and make several novel methodological contributions. These include a new evaluation corpus, a clever technique for creating the corpus, and an exhaustive approach to-wards sensitivity analysis for classification features.

Introduction

Tables matter. In a text document, they serve many purposes: they summarize, they aggregate, and they display change over time. The essence is this: a table provides for a compact and readable representation of relational or attributive information. In many text sources, in fact, the most important information is found in tables. This is certainly evident in financial domains, where tables of financial performance figures are the lingua franca of accountants and investors alike.

We are concerned in our work with applying information extraction and data mining techniques to these kinds of financial documents. In so doing, we have investigated an array of issues regarding tables, namely their identification, classification, and de-structuring. In the present paper, we address in particular the issue of classifying tables as they are found in HTML-based publications.

The reason we focus so specifically on HTML-formatted tables is of course because the World-Wide Web has changed everything about how we communicate. In our area of concern, corporate financials, companies increasingly publish both their annual reports and regulatory filings as web pages. The techniques we report on here are a direct outcome of this sea change in publication. While we focus in particular on financial documents, the work speaks to the issue of processing HTML tables in all domains.

Background: table processing

A substantial body of work has already addressed the question of processing tables in text documents. The earliest such work, which predates the widespread use of HTML, was naturally concerned with processing ASCII-formatted tables, meaning tables whose horizontal and vertical rules are drawn with hyphens and the like, and whose alignment is controlled by tabs and space characters. The primary challenge addressed by this work is that of de-structuring a table, meaning identifying its row and column headers, and delineating the content of its cells – see, for instance, Pinto *et al.* (2003), Hu *et al.* (2001), or Chen *et al.* (2000

With the advent of HTML, the table de-structuring task shifted to accommodate HTML's built-in table-creation constructs (which include header, row, and cell markup). This new problem isn't so much easier as it is richer. Wrapper induction, as this problem has come to be called (Kushmerick *et al.* 1997), has also proven a boon for the application of machine learning methods. Approaches range from rule-based supervised methods (Knoblock *et al.* 2003, Lerman *et al.* 2004, and many more) through generative unsupervised ones (Grenager *et al.* 2005).

In addition to recasting the problem of finding table structures, the introduction HTML tables has introduced a (surprisingly) more basic one: namely determining whether a table actually represents tabular data in the first place! The issue is that HTML's <TABLE> construct provides a convenient framework for much more than the creation of conventional numeric tables. In practice, <TABLE> is also widely used as a tabbing environment for the purpose of aligning columns of text, bullet symbols, footnote characters, and so forth. This is particularly true given the widespread use of page layout programs, as these often compile everything from bulleted lists to footnotes into <TABLE> constructs.

Wang and Hu (2002) consequently formulated the problem of identifying whether a table is *genuine*. By their definition, a genuine table is to be understood as a two-dimensional encoding of relational information, *e.g.*, bus schedules or stock market performance. A table is *non-*

Category	Description
Data v. time	Cell values are (mostly) numbers, rows are indexed to a quantity (e.g., income), columns are indexed to a date
Time v. data	Ibid, but rows are date-indexed, and columns quantity-indexed
Other num	Number-valued cells with no chronological indexing of either rows or columns
Text	Cell values are (mostly) text, meaning phrases, sentences, or even whole paragraphs

Table 1. Major table categories.

Category	Description
Numeric minor categories	
Income statement Balance sheet Cash flows	The three major accounting views of a company's financials
Consol. inc. stmt. Consol. bal. sh. Consol. cash fl.	Consolidated versions of the above, typically produced by a company's auditors
Stock info	Value, shares outstanding…
Pension plan Stock plan (esop)	Stock/pension plans often have their own financials
Misc. numeric	Anything else
Text-related minor categories	
Table of contents	For sections, attachments…
Bullets	Bulleted or numbered lists
Footnote	Often linked to num. tables
Signatures page	Auditor sign-off , etc.
Fat cats	Board members, executives
Stock info	Rare non-numeric tables
Text table	Tables such as this one
Formatting	Other alignment uses
Misc. text	Anything else

Table 2. Key minor table categories. Gray text shows low-count categories that were folded into other categories in most experiments.

genuine if it only presents a two-dimensional layout of unrelated elements that do not share an underlying relation such as time of arrival or price.

Cohen *et al.* (2003) follow this notion, and distinguish *true data tables* from other uses of the HTML <TABLE> tag. They cast the task of identifying true tables as a classification problem, and bring to bear the standard armamentarium of successful classification methods, among others: decision trees, Winnow, and maximum entropy models. Further, they note that table classification is of much more than just academic interest. Indeed, many table extraction methods based on wrapper induction depend critically on being presented the right kind of table. So to succeed in practice, the enterprise of wrapper induction depends keenly on solving this basic table classification problem.

Our approach here largely follows this path, but adapts it to the specific purpose of analyzing financial texts. Indeed, one of the key text sources in the financial area is corporate financial reports, in particular the annual 10-K filings submitted by publicly-traded companies in the United States. As these filings have increasingly been provided in HTML, the problem of distinguishing genuine tables has become correspondingly acute.

The need to make this distinction is several-fold. For one, financial data mining applications, *e.g.,* Chang *et al*, (2006), often provide interactive access to text based on a paragraph's subject classification. We therefore would like to provide an analogous access to tables through a fine-grained categorization of table content. In the case of financial reports, for instance, one might want a table of tables that distinguishes income statements, stock histories, and other like-minded accounting constructs. Part of our agenda is therefore to identify these distinctions through automatic classification methods.

In addition, many non-genuine tables contain text passages of significant value to financial analysis—the devil in the financial details often hides in footnotes. However, to productively apply our extraction and mining methods, we need to identify the format of these non-genuine tables, as listings of corporate officers, for instance, require a different treatment than do bulleted lists.

Finally, we also need to ensure that we not apply standard extraction methods to genuine financial tables. A name-tagged income statement is not a pretty sight, as financial tables often lack the contextual cues required for successful name tagging—even of money expressions.

The financial table classification task

The preceding discussion points to a need for two levels of table classification: a coarse distinction between genuine and non-genuine tables, as well as a fine-grained assignment of table type. Towards this end, we assign to a table both a major category and a minor category.

Table categories

To address the particulars of financial reports, we recognize four major categories (see Table 1). These broadly follow the division between genuine and non-genuine tables, where the first three categories taken together (the numeric categories) correspond to Wang and Hu's genuine tables. Text tables are our equivalent of non-genuine tables: for the most part they cannot comfortably be construed as encoding an underlying semantic relation. The special case of tables of contents, which could arguably be typed as either *text* or *other num* is, for our purposes, considered to be text-typed.

In addition to a table's major category, we recognize a range of minor categories, some of which are specific to numeric tables, and the others to text tables (see Table 2). For the numeric tables, the minor categories are defined on a primarily semantic basis, which is in keeping with our objective of extending subject classification from text to (genuine) tables. They mostly include such accounting categories as *income statement* and the like.

In contrast, the minor categories for text tables reflect our need to process specific table layouts in idiosyncratic ways. As such, their distinctions are primarily syntactic in nature. Among them is the particular category of *Formatting*, an undistinguished catch-all that captures the most egregious cases where <TABLE> is exploited for alignment purposes alone.

The table omits a number of particularly low-count categories, which we lumped into the two *Misc.* categories for most of our experiments. Among these rare tables were some that the annotators had marked as belonging to multiple minor categories, *e.g.,* financial tables that included both balance sheet and income statement information. In addition, rare examples of (*e.g.*) the *fat cats* minor category could arguably be seen as having either text or numeric major categories. We resolved these potential confusions by convention, so (*e.g.*) *fat cats* tables were always taken to be text-typed for their major category.

Corpus development

After setting guidelines for annotation, we created a corpus of 10-K filings in which all tables are marked with both their major and minor category. These documents vary a lot in length, though most are over 100 pages long. They also vary significantly in the number of tables they contain, ranging from a low of 22 tables per filing to a high of 363. This huge variance (avg= 110, sd= 94) is due to differences in reporting styles on the part of the filing corporations. In particular, some went to great length to provide multiple views of a particular financial table, broken down by geography, business, area, and so forth. Others provided only one such table, and yet others dispatched the bulk of their financial tables to their annual reports, and only cited them by reference.

For our purposes, this variance in reporting styles had implications relative to the practicality of using sequence models—more on this below.

For training and test purposes, we used 19 of these filings in the present study, with a total of 2,089 table instances. Most of our experiments defined a training-test split along document boundaries, collating enough 10-K's into the training camp to create a roughly 70-30 split, with 1,436 table instances for training and 633 for test.

We deliberately chose this particular approach to dividing training and test sets over the more conventional approach of N-fold cross-validation. Because our source documents have such inconsistent table density, cross-validation cannot be readily performed without splitting some documents up and assigning one part of their tables to training, and the other to test. This is not representative of actual operational conditions, where the "test" tables are always drawn from documents with no overlap with the training set. In this case, cross-validation measures are likely to overestimate actual runtime performance.

To construct the corpus, we used a clever trick that allowed us to mark up the filings *in situ* while reading them in a web browser. This gave us a WYSIWYG annotation capability, which we found critically necessary, since the source code for HTML tables is essentially unintelligible on its own. In our experience, efficiently and accurately judging the category of a table required that it be fully rendered by a browser. Wang and Hu (2002) describe a non-WYSIWYG annotation tool that would have been extremely cumbersome for our needs.

As to the clever trick, it worked as follows. We modified the HTML source code of our corpus documents, and added a pair of HTML <SELECT> menus in front of each HTML table. These pull-down menus allowed the annotator to indicate the major and minor categories of the table with no more than two mouse clicks. The whole body of the document was then wrapped in an HTML <FORM> with a SUBMIT button to save the mark up. A bit of HTML manipulation, plus a few external scripts was all that was required.

A preliminary round of annotation was performed by three project members, followed by a round of adjudication to resolve inter-annotator disagreements. This helped identify annotation tough nuts that required further guidelines. The full corpus was then annotated by the most consistently accurate annotator among the original three.

Experimental preparation

As noted, we cast the problem of identifying genuine financial tables as a classification task, and exploited a number of machine learning packages to learn a range of table classifiers. Prior to our experimental runs, we first followed a procedure involving the following three steps: preprocessing, feature extraction, and category mapping.

Preprocessing

As is often the case with HTML documents in the wild (on the Web), our corpus of 10-K filings proved unwieldy in their raw form. The issue is that they were difficult to parse due to unclosed tags and other factors. As such, a first preprocessing step converted the HTML documents to XHTML using the program TagSoup.[1] A second stage of preprocessing involved tokenizing the XHTML character data using a robust general-purpose tokenizer followed by some task-specific tokenization patch-up.

Feature extraction

We extracted a number of different classification features from the resulting XHTML documents. As with most previous work, we identified a number of structural features,

[1] Source: http://mercury.ccil.org/~cowan/XML/tagsoup/

such as the column or row count. In keeping with the standard text-classification literature, we also extracted bag-of-word lexical inventories. We also identified a number of features that attempted to model the lexical and structural context in which a table is found. Finally, we extracted a number of features that were aimed specifically at distinguishing some of our fine-grained categories, *e.g.,* counts of bullet-like or footnote-marking tokens, counts of date words or year numbers, and so forth. We then grouped these features into the following clusters, which we could independently activate or disable in any given experimental run.

Baseline. A baseline feature set was used in all experiments. It consists in part of bag-of-word counts of the tokens present in a table, as well as structural features, such as the number of rows and columns in the table as determined by the <TR> and <TD> HTML tags. We identified three versions of the baseline, depending on whether the bags of words contained lexemes in their original case, in their uncased forms, or both.

Column headers. Column header features were inferred from <TH> tags, when present, or by a heuristic that guessed potential column headers by examining whether the top two rows contained "real" alphabetic tokens (as opposed to only containing numbers). Headers were then modeled as both header-specific bags of words and a special feature that indicated whether headers were found and how the header bags were derived (by <TH> extraction, heuristically, and for some cases, both).

Row headers. These bag-of-word features selected the lexemes present in the row headers, *i.e.* the first non-empty column in a table (wholly-blank columns are often used for spacing purposes). This was done only for rows that were not deemed column headers.

Preceding table context. A binary feature indicating whether a table is immediately preceded by another table (modulo horizontal rules, page numbers, and the like).

Preceding lexical context. These bag-of-word features capture the *N* words that precede a table. We allowed for windows of either 0, 10, 50, or 100 words, each of which represented a separate experimental condition.

Preceding table type. This feature attempts to model sequencing effects that would be captured by sequence models such as generative (HMM) or discriminative (CRF) techniques. In practice, the classification assigned to the *nth* table in a document would be passed as the preceding table type feature for the *n+1th* table. For our experiments, however, we approximated this by simply passing in gold standard classifications as established by the annotators.

Following context. This group of features captures the context immediately following a table. The group consists of both the binary table context and the bag-of-words lexical context features that are separated for the case of preceding contexts. We lumped these together as preliminary experimentation showed that neither had especially significant effect on classification accuracy.

Section headers. These features attempt to model document section headers. Indeed, we had observed that in our corpus, certain tables occur more frequently in certain sections of the document than in others. To capture this effect, we attempted to heuristically identify section headers (via table-of-content lookup) and included their bag-of-word distributions in this feature group.

Task-specific tokenization. This feature group, when activated, invoked an extra round of tokenization, in particular for number sequences that would otherwise be atomized, *e.g.,* "(1.1)" "15 (c)" and the like. It also introduces a set of features that separately count the number of tokens that could be used as bullets, the number that could be used as footnote markers, and the number that could be used as table-of-contents entries. Note that tokens like "(1)" could be used for all three of these purposes.

Date and number normalization. A group of features that counts the number of date words, year numbers, and plain numbers in the table. This effectively normalizes these constructs, as if they were being replaced by a special DATE or NUMBER token.

Bottom lines. We attempted to implement a "bottom line" detector, as this can be a strong lexical cue for numeric tables. This was done by finding the bottom-most row header in a table, and creating separate bag-of-word features for this entry.

Category mapping

The result of all this feature extraction was to produce a collection of feature vectors labeled with the annotator's judgment of their major and minor categories. For actual experimentation, these categories were then remapped. The purpose of doing so stage is two-fold.

First, as our original repertoire of minor text categories makes an impractically large set of distinctions (39, not all of which are shown Table 2), we needed to reduce their number to a more manageable level. As noted earlier, we mapped a number of similar low-count categories to each other, producing a collapsed set of categories.

Second, we also took advantage of the category-remapping procedure to evaluate a range of possible use cases that so not require identifying the full set of minor categories. The cases we considered were as follows.

Plain (39 categories). The labels consisted of the original categories and sub-categories in the annotated data.

Collapsed (15 categories). A first round of category mapping that collapsed low-count minor categories.

Numeric + collapsed text (8 categories). A further round that collapses all the numeric minor categories together. This corresponds to a text processing use case where one might ignore the numeric tables, but want to (*e.g.*) name-tag text in non-genuine (text) tables.

Major category only (4 categories). All the minor categories were ignored, and only the four major categories were considered.

Numeric vs text (2 categories). The three number-oriented major categories were further collapsed together. This condition effectively corresponds to making the same genuine *vs.* non-genuine distinction in previous work.

Experimental results

The bulk of our experiments were performed with a multinomial maximum entropy classifier, using a Gaussian prior of 100.0 for all runs. These are the results we report below. We additionally repeated all these experiments with the LibSVM implementation of support vector machines, but were unable to obtain comparably good results. This should not be taken as the final word about SVMs on this task, as time limits prevented us from performing an exhaustive analysis of the hyper-parameter space. Finally, we also tried some comparison experiments with conditional random field classifiers. We had hoped to see some sequence effects from these experiments, as mutual information measures show significant predictiveness between sequential pairs of tables. However, CRFs also underperformed maximum entropy models.

First experiments

A first round of experiments yielded the following results for the five use cases under consideration.

Use case	Accuracy
Plain	71.72%
Collapsed	76.30%
Numeric + collapsed Text	89.73%
Major category	92.26%
Numeric vs. text	98.10%

These figures report the highest-performing classifier that was learned for each of these five cases. For each use case, a large number of classifiers were learned, based on different configurations of activated or disabled feature groups. We were encouraged by these performance figures, especially since the first three represent decisions between a relatively large number of classification labels (39, 15, and 8 respectively).

Nonetheless, these figures do not tell the whole story. When probing the range of training configurations, we found surprisingly large variance between closely related configurations of features. For example, the winning 15-way classifier for the *Collapsed* use case relies on both case and uncased bag-of-word entries. However, switching to a cased-only version of the bags of words, causes performance to drop by 5 points. To understand this variance, we engaged in a comprehensive analysis of training configurations.

Comprehensive configuration analysis

This analysis explores the entire search space defined by the clusters of features we implemented for this task. That is, we treated each of our $\varphi_1\ \varphi_2 \ldots\ \varphi_n$ feature clusters as a term in a giant $\varphi_1 \times \varphi_2 \times \ldots \times \varphi_n$ cross product, and independently activated or disabled each cluster so as to produce all k^n different configurations of feature clusters. (Note that $k \geq 2$, as some of these clusters are either on or off, and some have several multiple configurations). For our first round of experiments, this amounts to ±4,000 configurations, *i.e.*, 4,000 sets of training/test vectors.

We then trained classifiers for each of these configurations, a process that would be wholly impractical in most instances. We were fortunate enough, however, to have access to a grid of twenty-six Apple Xserve computers. On this grid, training these several thousand classifiers was an overnight process that took on the order of 12-18 hours. The thousands of resulting evaluation scores were then entered into a relational database, which greatly simplified the subsequent process of analyzing configurations and identifying the best-performing ones. The performance results we report here were obtained in this way.

Observations

We used these configuration analyses to drop some features that proved to be poor performers in our first round of experiments, and to re-engineer and add others, resulting in the feature clusters that we report here. We also performed a range of error-term analyses based of the confusion matrices for high-performing configurations. Two trends emerged in particular.

First, among *genuine* financial tables, the most confusions were between income statement tables and stock info tables. We suspected that this was due to a peculiarity of our guidelines that required certain very short income tables to be classified as stock info. On reflection, this requirement seemed poorly conceived, so we re-annotated those tables as income statements. Income statements were also prone to be misclassified as *Misc* and vice-versa. We suspected that this too was an experimental artifact, as we had chosen to map certain multi-category tables to *Misc*, as their counts were generally low. Many of these tables, however, contained an income statement coupled to a balance sheet, so for subsequent experiments, we remapped them to the *Income statement* category.

A second trend in our first experiments, is that among *non-genuine* (text) tables, the largest number of confusions were between bulleted text, footnotes, and to a lesser degree tables of content. This observation caused us to introduce a number of changes to the way bullet and footnote characters were tokenized, as well as to the way we calculated table adjacency.

Finally, we decided to annotate a few more 10-K filings, in the hope that larger numbers of training instances would mean less variance between experimental conditions.

Second experiments

We performed a second round of experiments incorporating the engineering changes to the feature selection and category mapping. These experiments also included the changes we made to the training data as well as the newly-annotated 10-Ks (for an additional 231 training instances). Because of time considerations, we did not train or evaluate the 39-way classifier for the Plain use case. As before, we performed a brute force exploration of the feature space, yielding the following high-performing classifiers.

Use case	Accuracy	Error Δ
Collapsed	81.74%	-23%
Numeric + collapsed Text	92.39%	-26%
Major category	92.39%	-1.7%
Numeric vs. text	98.48%	-20%

Clearly, the engineering and re-annotation effort paid off in terms of significant reductions in the error term for three out of the four use cases. The final results also stand strongly on their own, as accuracies of 81% and 92% are considered very good for 15-way and 8-way text categorizations tasks (respectively).

Discussion

We would particularly like to point out our final result for the Numeric vs. Text use case, as this is our closest point of comparison with previously published work. For instance, Wang and Hu (2002) report a best F of 95.89 for their genuine *vs.* non-genuine classification task. Cohen et al report a best F of 95.9 for their task of finding true data tables. There are of course significant differences between our financial tables task and those attempted by these other authors, so their results and ours are not wholly comparable, but we would advance that our performance at this binary classification task is within the state of the art.

As with our first round of experiments, we analyzed the contribution of our various feature clusters to the performance of the highest-ranked classifiers. We were pleased to note that all four of the classifiers used substantially the same set of features. They all used the cased and uncased bags of words, row headers, and some variant of the heuristic column headers. They all tested for the presence of a preceding table, and were all able to exploit the preceding table's categorization. For the most part (3 configurations out of 4), they used 10 or fewer words of lexical context, and did not bother to normalize dates or numbers. They were split evenly as to whether to use the tokenization feature cluster.

One point worth noting is that these results were all obtained using gold standard labels for the contextually preceding table's category. In a running implementation, these would have to be filled in by the results of previous classifications. To assess the degree to which performance may be impaired without gold standard table contexts, we considered the feature configurations that were identical to our winning ones, but with the preceding table type feature turned off (a simple database lookup, given our comprehensive analysis). As expected performance is reduced without the preceding table class, but only slightly.

Use case	Accuracy w/ type	Accuracy w/out type
Collapsed	81.74%	81.33%
Numeric + collapsed Text	92.39%	90.32%
Major category	92.39%	92.26%
Numeric vs. text	98.48%	98.34%

Many more analyses of this kind remain possible. One of the most interesting aspects of this work, however, is that our comprehensive (brute force) exploration of configurations makes it possible to actually ask this kind of question and answer it quickly. With the growing availability of grid computing, we expect this technique to catch on and prove highly valuable to practitioners of machine learning.

We also hope that other researchers will be drawn to the financial tables classification task, what with its multiple levels of classification and its domain intricacies. We look forward to the dialogue that we hope will follow.

References

Chen, H, Tsai, S, and Tsai, J. 2000. Mining tables from large scale HTML texts. In *Proc. of the 18th Int. Conf. on Computational Linguistics (COLING 2000)* , pp. 166-172.

Cohen, W, Hurst, M, & Jensen, L. (2003): A Flexible Learning System for Wrapping Tables and Lists in HTML Documents. In Antonacopoulos, A, & Hu, J. (eds.) *Web Document Analysis: Challenges and Opportunities*, World Scientific Publishing.

Grenager, T, Klein, D, & Manning C. 2005. Unsupervised learning of field segmentation models for information extraction. In *Proc. of ACL 2005,* pp. 371-378, Ann Arbor.

Hu, J, Kashi, R, Lolpresti, G, & Wilfon, G. 2001. Table Structure Recognition and Its Evaluation, In *Proc. Document Recognition and Retrieval VIII*, pp. 44-55.

Hurst, M. 2001. Layout and language: Challenges for table understanding on the web. In *Proc. 1st Intl. Wkshp. on Web Document Analysis*, pp. 27-30, Seattle, WA.

Knoblock, C, Lerman, K, Minton, S, & Muslea, I. 2003. Accurately and reliably extracting data from the web: A machine learning approach, In Szczepaniak, P, Segovia, J, Kacprzyk, J, & Zadeh, L (eds.) *Intelligent Exploration of the Web*, Springer-Verlag.

Kushmerick, N, Weld D, & Doorenbos, R. 1997. Wrapper induction for information extraction, In *Proc. of IJCAI-97*.

Lerman, K, Getoot, L, Minton, S, & Knoblock, C. 2004. Using the structure of web sites for automatic segmentation of tables. In *Proc. of ACM SIG on Management of Data (SIGMOD-2004)*.

Pinto, D, McCallum, A, Wei, X, & Croft, W. B. 2003. Table Extraction Using Conditional Random Fields. In *Proceedings of the 2003 ACM SIGIR Conference*.

Wang, Y. and Hu, J. 2002 A machine learning based approach for table detection on the Web. In *Proc. of the WWW 2002 Conf*erence.

Yoshida, M, Torisawa, K, & Tsujii, J. 2001. A method to integrate tables of the world wide web. In *Proc. 1st Intl. Wkshp. on Web Document Analysis*, pp. 31-34.