On Table Extraction from Text Sources with Markups

Lorenz Weizsäcker   Johannes Fürnkranz
Technische Universität Darmstadt
{lorenz,juffi}@ke.tu-darmstadt.de

Abstract

Table extraction is the task of locating tables in documents and extracting their entries along with the arrangement of the entries inside the tables. The notion of tables applied in this work excludes any sort of meta data, e.g. only the content elements of the tables are to be extracted. We follow a simple unsupervised approach by selecting the tables according to a score that measures the in-column consistency as pairwise similarities of entries where separators columns are also taken into account. Since the average is less reliable for smaller table this score demands a levelling in favor of greater tables for which we make different propositions that are covered by experiments on a test set of HTML documents. In order to reduce the number of candidate tables we use assumptions on the entry borders in terms of markup tags. They only hold for a part of the test set but allow us to evaluate any potential table without referring to the HTML syntax. The experiments show that the discriminative power of the in-column similarities are limited but also considerable given the simplicity of the applied similarity functions.

1 Introduction

This work is about table extraction on HTML-documents or on other source types that use mark-up-tags and can easily be transformed into a sequence of ASCII symbols without impeding the elicitation of tables. As extraction output we target the entries along with their arrangement in row and columns letting aside any sort of meta data.

Following [6],[3] the task of *table extraction* strips downs to two subtasks. For solving the task of *table location* we have to find the substrings of the input string that contains the tables. For *table recognition* we want to get the tables content and structure out of those substring having the input string at hand. A substrings is here defined by both, the sequence of character it contains and its *slice*, the start and the end position, on the string it is taken from.

In principle, mark-up-languages allow addressing tables with tools for querying the structure induce by the markups. However, for neither of the two subtasks we can fully rely on the markups. For instance, in HTML-documents *table*-nodes often are used for arbitrary layout porpuses. Further, the location task also has a significant genuine error in the sense that for some potential tables it is a matter of tast wether we takes them as extractioin targets or not. The correct output is not defined but by the choosen example set. We do not intend to extract meta data such as titles, column headers, separating rows that are empty or contain titles of the subsequent table part. Here again, for many types of meta there are special node definitions but these are not always used. For instance, we have often to deal with column headers which take, according the markups, the place of first data row of the table.

For the approach presented here we do not consider the definitions of the mark-up-languages. Instead, we confine ourself to inspecting the statistical structure inside a column of a table, where column refers to columns of table entries and columns of entry separators as well. More precisely, we measure the consistency of a column by the pairwise similarity of its elements, where similarity refers to a plug-in string kernel. The basic observation is that the entries in a given column of a table tend to be more similar to each other than to other potential table entries in the document [14],[2]. With this report we want to frame out an extraction algorithm that uses the in-column similarities as single extraction criterion and see to what extend the extraction of tables can draw thereon.

Our algorithm intends to consider all potential tables within the input string likewise searching substrings with certain properties. Without further restriction of the output space, this is too expensive since we do not only have to find the substrings of the input covering the tables but also determine therein all substrings representing the entries. If we regard any position in the string after a given position as candidate start point of the next entry, there are far too many candidate tables. For this reason, we consider documents with mark-up tags such HTML documents. Obviously, the search space for these documents is smaller since the border of a table entry can be assumed to concur with the border of a tag.

Unfortunately, real world documents contain thousands of tags such that the search space still is to big for inspecting it element by element. This means that we either have to use non-exhaustive search, e.g. we evaluate the extraction criterion only for some elements in the space and find or hope to find the most promising candidate nevertheless. Or, we try to further reduce the search space by means of stronger assumptions on the entry borders. We have chosen latter option by applying the assumptions given in section 3 which shrink the size of the space of candidate outputs to a moderate size. The drawback of this choice, as reported in the experiments section 5, is that the assumptions hold only for a minority of the documents in the data set we used. We decided to take this loss in relevance of the results because at first we wanted elaborate the extraction criterion itself.

And indeed, there is a principle catch with extraction criteria such as the criterion specified in section 4.2 and 4.3. We intend to evaluate a candidate table by the average in-column similarity of the table entries. However, the reliability of this average score strongly suffers when the candidate table is small and we therefor average over a small number of similarities only. We respond to this problem by deliberately decreasing the chances of smaller candidate table, an procedure to which we refer to as *levelling*.

The outline of this paper is the following. An unsorted review on related work is given in section 2. Section 3 provides notation and specifies the assumptions for the reduction of the search space. The extraction criterion, called table score, is given in section 4 where we also provide different proposals for levelling schemes. In section 5 we report on experimental results based on the set data set from [14] and section 6 contains conclusions and depicts further work.

2 Related Work

2.1 Binary Classification with Content Features

The reference most related to this work is [14]. The authors reduce the table location task to a binary classification problem. First, the leaf table nodes are marked as candidate target tables. Then, each candidate table is mapped to a feature vector that in turn is the input to a classifier which decides whether the vector represents a target table or not. The feature vector consists of the feature groups for words, layout and content respectively.

For *word features* a bag of words is collected from the text nodes of each candidate table and mapped to a feature vector in a TF.IDF fashion with respect to the set of positive and negative candidate tables in the example set.

The *layout features* are based on counts of rows, columns, cells (leaf text nodes) and the lengths of the cells. Here, rows refer to nodes declared as table rows by HTML-tags. We have not yet understood to what columns refer to in cases where the rows differ in the number of entries. The standard deviation of the number of cells per row is one of the layout features. All normalization are *local* in the sense that they refer to corresponding elements inside the same table.

Another layout feature is called *cumulative length consistency*. It is intended to capture the observation that the lengths of entries for a given column are often similar. The *content-features* transfer this concept to consistency with respect to content types such as *hyperlink, image, alphabetical*. This concept of consistency has also been the initial motivation for our work. While in our work this idea is formalized in terms of pairwise similarities, the consistencies in [14] are computed based on a reference value that is the mean for the lengths and the most frequent value for types. It might be interesting to draw a clear sight on the relation of these two formalizations, but we have not done this.

2.2 Display Models

In order to ease the reading of web pages, the developers of browser invest much effort to translate HTML-files into a clear arrangement one a 2-dimensional display. In [3], see also [4], the authors propose to make use of this arrangement result as provided by the mozilla rendering engine for solving the task of table extraction. We estimate this as a particularly clever approach. Tables can be characterized as boxes that are tiled in a regular manner fulfilling certain constraints. Though the arrangement pattern are of comprehensible complexity the approach yields good results on a large set of web-pages. It works out without any training.

The approach of [4] and the approach presented here are complementary in sense that the first focuses on topological structure of the input without considering the content of the table while the latter inspects the possible entries and separators without explicitly modeling the meaning of the separators for the arrangement. Nonetheless, they also overlap to some extend. Certainly, string similarity of separators is correlated to similar local topology. Also, in [4] congruence in text and background colors is taken into account which has no influence on the topology but on the similarity measures applied here.

2.3 Belief Propagation on Line Sequences

Pinto et. al. consider the problem of table extraction as segmentation of the input string into segments carrying labels such as *table-title lines, data-row lines, non-table lines* [10]. The authors target plain input strings and apply the assumption that segment borders are at line breaks such that entire lines can be used as tokens for linear belief propagation. Under a conditional random field model (CRF) this yields good results on set of documents fulfilling the assumption even with a rather simples feature set. In the provided state, the approach targets the location of the tables and meta-information on its rows but does not reveal the row-column structure of a table. To fix that the authors proposed to apply a 2-dimensional CRF with characters as token. For such a model, however, the optimization is difficult [13].

2.4 What is not Discussed

In this work we do not aim to extract meta-data as table titles or headers ([10]) nor provide additional tags carrying information that could used for further processing of the extracted data. The problem of integration of the extracted data [11] is not considered either, though this work is much inspired by [5] where data integration is a main application.

3 Notation and Assumptions

In this section we want get a more formal grip on what the candidate outputs of table extraction are. First, we formally specify the general output space independent of concrete ground truths and extractors. Then, we formulate assumption made by the table extractor given in section 4. These assumptions only hold for less than one out of six input-output examples from the ground truth we build in section 5 but they allow us to use a reduced output space that is suited to study table extraction by column-wise similarities with reduced technical outlay.

3.1 Output Formalization

For table extraction we want to map an *input string* x over a fixed alphabet Σ to a *list of tables* $y = (t^1, \ldots, t^q)$ of indefinite length q. The k-th *table* in that list is a list of rows $t^k = (r^1, \ldots, r^{n_k})$ where the i-th *row* in turn consists of $m_{k,i}$ entries $r^i = (e^1, \ldots, e^{m_{k,i}})$. An *entry* is a string over Σ.

If $m_{k,i} = m_{k,\bar{i}}$ for all $1 \leq i, \bar{i} \leq n_k$, we say the table t_k is *rectangular*. In this case the tuples $(c^1, \ldots, c_k^m)$ where $c^j = (e_j^1, \ldots, e_j^{n_k})$ such that $r^i = (e_1^i, \ldots, e_{m_k}^i)$ are referred to as *columns* of t_k.

Instead of defining entries as self-contained strings, one can also represent them as substrings or only as slices on the input string [7]. The above representation has the advantages that it is more readable and that it is decoupled from the input and therefore more robust. Nonetheless, we assume that the entries occur in the input x as non-overlapping substrings in the order induced by the table structure (depth first). When we talk about an entry in the following section, we may refer to its proper string, its slice or both, depending on the context.

3.2 Assumptions

In the following we formulate some strong assumptions on the input string and the tables therein. These assumptions characterize the cases to which the proposed approach presented in this paper is restricted to.

The general assumption is that the input string can unambiguously be segmented into an *alternating segmentation* of *tag segments* and *content segments*. The content segments are potential entries while tags or groups of tags form potential separators between entries.

More specially, we assume that table entries always contain at least one non-whitespace character and the markups are given as non-overlapping *tags*, substring that start with < and end with > but have neither of both in-between. This way we can define a tag segment as sequences of tags only separated by whitespace. The content segments are segment between the tag segments.

One major demerit in these assumptions is that table entries can contain tags. But mostly they have formating purpose and surround a single content segment. Let the *peeling function* $\gamma : \Sigma^* \rightarrow \Sigma^*$ take away any prefix and suffix of its string argument that consist of tags and whitespace only. We assume that any table entry, when peeled by γ, is a content segment. Of course, entries may contain tags surrounded by content segments. In such cases the assumptions do not hold and the algorithm below will fail.

The alternating segmentation is denoted by G and the separated subsequences of G containing separator segments and entry segments only by G^{s} and G^{e} respectively.

$$G = (g_1^{\mathrm{s}}, g_1^{\mathrm{e}}, g_2^{\mathrm{s}}, g_2^{\mathrm{e}}, \ldots, g_p^{\mathrm{s}}, g_p^{\mathrm{e}}),$$
$$G^{\mathrm{s}} = (g_1^{\mathrm{s}}, \ldots, g_p^{\mathrm{s}}),\ G^{\mathrm{e}} = (g_1^{\mathrm{s}}, \ldots, g_p^{\mathrm{s}}) \quad (1)$$

If the assumption does hold, the extraction of the table entries reduces to the selection of a subsequence of G^{e}. We further restrict the output space by additionally assuming that tables consist of consecutive content segments. This implies that there are no separating rows, which does not hold for a rather big fraction of tables. But this additional assumption reduces the space of candidate tables to a level that permits exhaustive search.

3.3 Final Output Space

Applying the above assumption, we can specify the output space for given input x in very simple terms. If the alternating segmentation of x has length p, any candidate table within x is can be represented as a triple (a, m, n) where a is the index of its first entry in G^{e}, m the number of columns, and n the number of rows such that $a + mn - 1 \leq p$. Let us denote this space of output tables by $T(G)$.

How many table does $T(G)$ contain? Let $\mathrm{n}^{\mathrm{p}}(a)$ be the number of tables starting at position a, and $\mathrm{n}^{\mathrm{c}}(l)$ the number of tables that can be built from at most l consecutive content segments in G^{e}.

$$|T(G)| = \sum_{a=0}^{p-1} \mathrm{n}^{\mathrm{p}}(a) = \sum_{l=1}^{p} \mathrm{n}^{\mathrm{c}}(l) \quad (2)$$

The term $\mathrm{n}^{\mathrm{c}}(l)$ equals the number of ordered pairs that multiply to at most l. On limit average it grows as $l \ln l$ as can be seen from the following bounds that hold for any $l \in \mathbb{N} \setminus \{0\}$.

$$l(\ln l - 1) - 2\sqrt{l} \leq \mathrm{n}^{\mathrm{c}}(l) \leq l(\ln l + 2) + 2\sqrt{l} \quad (3)$$

The number of candidate tables is therefore bounded from above by $p^2(\ln p + 1)$.

4 Extraction by Column-Wise Similarities

We want to study a simple score based table extraction algorithm. The algorithm is named *CSE*, standing for *column-wise similarity evaluation*, as its selects the output tables according to a table score based on the in-column similarities.

4.1 Iterative Maximization

The proposed table extraction algorithm tries to solve the table location and recognition task in one go by maximizing a *table score* H over the candidate table in T which will be given below. The maximizer $\hat{t}$ of H is one table in the output list $\hat{y}$.

$$\hat{t} = \underset{(a,m,n) \in T(G)}{\operatorname{argmax}} H(a, m, n) \quad (4)$$

The other tables in $\hat{y}$ are obtained by incrementally exclude candidate tables that overlap with the tables extracted so far and retrieve the maximizer from the remaining ones. The output list is finally sorted by the position of the first entry a such that the tables appear in the same order as they do in the input string.

We want to assume that true number k of tables is given in addition to the input string. That is, we let aside the problem of detecting where in the sequence of maximizers the entries stop to be tables. In case that the first $l < k$ table already cover the sequence such that no further table can be extracted, the remaining tables are defined as empty tables that are tables with zero rows.

We refer to an input hint such as the number of tables above as *promise* using a notion from complexity theory [1]. More generally, a promise is a string that is added to the input and represents certain information on the true output.

4.2 Table Score

The table score should express how table-like a table is. It is exclusively obtained from scores on the columns of the table, where column here means both, column of entries and column of separator between entries and rows. Row separators are treaded like entry separators. The difference is that in a table there is one row separator less than there are separators between two content columns because we model a table to start with its first entry and to end with its last entry.

A table column is represented by a tuple (u, b, m, n) where $u \in \{\mathrm{e}, \mathrm{s}\}$ is its type (content- or separator-segment), b is the index of the first entry of the column in G^u, n is the number of its entries and m the number of columns the table to that the column belongs to has. We denote the score of a column (u, b, m, n) which should express its column-likeness by $h^u(b, m, n)$.

For a given table $(a, m, n) \in T(G)$ the table score is the sum of scores of its columns divided by a normalization term $z(m,n)$.

$$H(a,m,n) = \frac{1}{z(m,n)}(h^{\mathrm{e}}(a,m,n) + \sum_{j=1}^{m-1}(h^{\mathrm{s}}(a+j,m,n) + h^{\mathrm{e}}(a+j,m,n)) + h^{\mathrm{s}}(a+m,m,n-1)) \quad (5)$$

For either type u we aim the score of a column (u,b,m,n) to give a measure of how well the elements of the column $(g_{a+im})_{i=0,\ldots n-1}$ in G^{u} fit together. We can model this by the sum of their pairwise similarities. Let $\bar{s}^u : Q^u \times Q^u \to [0,1]$ be a similarity measure where Q^u is the set of possible segments of type u. Then, the score of a column (u,b,m,n) is given by

$$h^u(b,m,n) = \sum_{0\le i<j<n} \bar{s}^u(g_{b+im}, g_{b+jm}). \quad (6)$$

The normalize term is the total number of similarities that are taken into account

$$z(m,n) = (2m-1)\binom{n}{2} + \binom{n-1}{2} \quad (7)$$

such that H is the average similarity between entries or separators stemming from the same column.

4.3 Entry Similarities

A good design of the similarity functions s^{e} and s^{s} is an important factor for the performance of CSE extraction algorithm. This can be seen by trying different similarity functions and/or different parameters of these. We did not undertake systematic studies on this and in this report we neither want to discuss what adequate similarities may be nor to what extend one can expect a good choice of similarities can evaluate the performance. In the following we briefly describe the similarities we applied in our experiments as reported in section 5.

To make sure that a similarity is in the range of 0 to 1 regardless the actual similarity function s^u, the CSE algorithm always uses the normalized variant $\bar{s}^u$.

$$\bar{s}^u(g,\bar{g}) = \frac{s^u(g,\bar{g})}{\sqrt{s^u(g,g)s^u(\bar{g},\bar{g})}} \quad (8)$$

For both segment types with apply a similarity s^{l} based on the segment lengths. Let us write $|x|$ for the length of a string x. The length similarity of two segments a and b evaluates the ratio of the greater length to the smaller one through an exponential decay.

$$s^{\mathrm{l}}(a,b) = \exp(-g(a,b)), \quad g(a,b) = \frac{1+\max(|a|,|b|)}{1+\min(|a|,|b|)} - 1 \quad (9)$$

While the similarity of separator segments is reduced to the length similarity, e.g. $s^{\mathrm{s}} = s^{\mathrm{l}}$, the similarity on entry segments additionally checks whether the two string are of the some type where the type of string is either *integer*, *non-integer number*, or *other*. This type similarity $s^{\mathrm{t}}(a,b)$ is 1 if the types of a and b match, and 0 otherwise. The entry similarity is given as product of length and type similarity: $s^{\mathrm{e}}(a,b) = s^{\mathrm{t}}(a,b)s^{\mathrm{l}}(a,b)$.

In principle we can plug-in any string kernel as similarity function on segments, but it should be noted that the evaluation of the similarities must have moderate costs because any two segments of the same type are to be compared.

4.4 Score Levelling

Unfortunately, simply averaging over the relevant similarities is not an option because of two reasons. To the first we refer as *subtable problem*. Consider two tables, where one table is a subtable of the other, both having approximately the same elevated score. In such a case we prefer the greater table, because we assume that a wrong extension of a complete table decreases the score while false dropping of rows does not so.

The second issue is the problem of *winning by variance*. The smaller the table shape, e.g. the less candidate entries are taken into account, the less reliable is the table score as selection criterion. The distribution of scores is less concentrated for smaller shapes than for larger ones because we average over fewer similarities and because the candidate tables have less average overlap to each other. Also, there are more non-overlapping candidates in absolute numbers. These properties make them more likely to erroneously exceed the score of the correct target tables.

The two issues differ in their scope. The subtable problem refers to preferences among certain outputs having similar score H. In other settings we might use a similar score although no or other preferences exists. In contrast, the winning by variance problem is not related to the input-output distribution but refers to the different predictive qualities of the scores due to the score structure.

The framework to approach these issues is to apply a *score levelling* that maps the score H of a candidate table to a leveled score $\bar{H}$. The levelling *increasing with the shape* that is the original score is decreased the more the smaller m and n are. We confine ourself to *linear levellings* that have the form below. For better reading, a single term s is used to denote the shape (m,n).

$$\bar{H}_G(a,s) = \frac{H_G(a,s) + b_G(s)}{c_G(s)} \quad (10)$$

One may try a reasonable guess for c and b and use it as *ad hoc levelling* or fit the levelling from a broader class of functions using a training set. In the subsequent subsections we discuss instead ways to tackle the problem of winning by variance by levellings of the form (10) directly. In contrast, we do not respond to the subtable problem explicitly. We assume that all levellings that do not increase to slowly will sufficiently solve it. If the subtable is much smaller, the the score of the supertable is decreased much less. If, on the other hand, the subtable only misses a few rows, then its score is unlikely to be much greater since it contributes the main part to the score of the supertable.

In the following we try to give a better idea of the levelling approach and therefore use simplified setting: we assume the input-output pairs $Z = (G,K)$ are drawn from a distribution P which only supports segmentations G that have a given length p and contain exactly one true table denoted by K. We say P has *input length* p and *output length* 1.

Let l be a *score loss* on candidate tables, for instance the 01-loss $l_Z(t) = [\![H_G(t) > H_G(K)]\!]$ or the *hinge-loss* $l_Z(t) = \max(0, H_G(t) - H_G(K))$. The *risk of the score* $\mathrm{R}_P(H)$ is the expected sum of losses over all candidate table in $T(G)$.

$$\mathrm{R}_P(H) = \mathrm{E}_{Z\sim P}\sum_{t\in T(G)} l_Z(t) \quad (11)$$

We want to decompose that risk along the shapes. Let S_p be the set of *feasible shapes* and $A_p(s)$ the set of *feasible* *positions* given the shape s and let $e(s) = \mathrm{E}_{G\sim P}(e_G(s))$ with $e_G(s) = \sum_{a\in A_p(s)} l_Z((a,s))$ be the *risk at shape* s.

$$\mathrm{R}_P(H) = \sum_{s\in S_p} e(s). \tag{12}$$

The approach of shape levelling assumes that independently of P but due to the structure of H the risk e is greater for certain shapes such that reducing the chances of a such an s reduces $e(s)$ more than it increases the risk at other shapes and therefore leads to a smaller total risk which in turn is assumed to correspond to a better extraction performance.

Note that we do not further discuss that notion of risk nor the choice of the underlying score loss. Here, they are only used in order to explain the concept of levelling but they are not directly taken into account when we discuss ways to define levellings below.

Fair Levelling

The idea of *fair levelling* is to design the levelling such that the score maximization scheme does not favor any shapes when the segmentation can be assumed to contain no tables.

Let P be of input length p and output length 0. The segmentations drawn according to P do not contain any tables and we therefor call to such a P *table-less*. Given a table-less distribution P, we say that the table score is *fair with respect to* P if the shape s^* of the maximizer of H_G has uniform distribution over S_p.

A sufficient condition for obtaining a fair score is that that distribution of the maximum does not dependent on the shape. Since this goal is overstated, we propose a rough approximation thereof by setting the expectation of the maxima for different shapes to one level. Let $\mu_p = \mathrm{E}_{G\sim P}\mu_G$ be the expected average score over S_p and $A_p(s)$ and let $\nu_p(s) = \mathrm{E}_{G\sim P}\nu_G(s)$ be the expected maximum of the scores over $A_p(s)$.

$$\mu_G = \operatorname*{avg}_{s\in S_p, a\in A_p(s)} H_G(a,s) \quad \nu_G(s) = \max_{a\in A_p(s)} H_G(a,s) \tag{13}$$

With the following levelling, we approximate a fair levelling by standardizing the expected maxima given the shape, e.g. we set $\mathrm{E}_{G\sim P}\max_a \bar{H}_G(a,s) = 1$ for all s in S_p.

$$\bar{H}_G(a,s) = \frac{H_G(a,s) - \mu_G}{\nu_p(s) - \mu_p} \tag{14}$$

In praxis, we have to use estimations of μ_p, $\nu_p(s)$ that we obtain in our experiments in section 5 simply by averaging $\nu_G(s)$ and μ_G over a set of segmentations drawn from some P. The term $-\mu_G$ in the nominator in (14) is irrelevant for the maximization but it accents the standardization character of the levelling.

Table-Less Models

The approach of approximated fair levelling demands that we have a table-less distribution P, to which we now refer as *segmentation model*, at hand. We discuss a few simple table-less models.

The first model, called *Bernoulli model*, is a simply *iid model* where we draw the segments independently and with equal chances from $\{0,1\}$. The similarity of two segments is 1 if they are equal and 0 otherwise. This model has little to do with the segmentation from which we want to extract tables but still might be sufficient to design a effective levelling as it does capture the structure of the table scores.

The second model, which is named *shuffling model*, is an iid model as well. A segmentation is drawn from an example set and then we sample the segments for the new segmentation according to the distribution of segments in the sampled segmentation. At least with high probability we can assume that we do not find any table in a segmentation that is drawn according to either of these iid models.

Last, we consider the *empirical model* where random draw from a set of example segmentations containing no tables is taken are empirical distribution. From one segmentation of length p we obtain sample value of $\nu_p(s)$ for any $s \in S_p$. But contrary to the iid models, we only get empirical evidence for some p and therefor need a more elaborate smoothing technique than for the iid models where we can generate segmentation for any p. On the other hand, we assume the levelling to be monotone in each of its the arguments m, n and p what strongly decreases the data demand. Nonetheless, the definition of such a smoothing remains future work.

The empirical model as as well as the shuffling model amount, strictly speaking, to supervised table extraction since we only want to sample segmentation not containing tables. On the other hand, this binary labeling is very cheap compared to the actual extraction of tables.

Variance Levelling

A simple variant of the fair levelling is *variance levelling* where we standardize the score in the classical sense with respect to some table-less distribution P. That is, we divide the score by the standard deviation that the tables of shape s are expected to have when we run over the feasible positions.

$$c(s)^2 = \mathrm{E}_{G\sim P} \operatorname*{avg}_a (H_G(a,s) - \mu_G)^2 \tag{15}$$

With an iid model we can explicitly compute the values $c(s)$ from two parameters of the underlying segment distribution. For simplicity, we now include the separator subsequent to the last entry segment into the table such the table score is the sum of $2m$ independent identically distributed columns scores. Let H be the score of some candidate table in $G \sim P$ with shape (m,n) and let C be the score of a column in G having n entries. Taking the column score as U-statistic for a binary kernel, we have

$$\begin{aligned} \mathrm{V}(H) &= \frac{1}{2m}\mathrm{V}(C) && (16) \\ &= \frac{1}{mn(n-1)}\left((n-2)\sigma_1^2 + \sigma_2^2\right) && (17) \end{aligned}$$

where $\sigma_1^2 = \mathrm{V}_X(\mathrm{E}_Y(s(X,Y)))$ and $\sigma_2^2 = \mathrm{V}_{X,Y}(s(X,Y))$, see for instance [8]. For the Bernoulli model the parameters σ_2^1 and σ_2^2 can easily obtained from success probability q. In case of the shuffled model we have to estimate them by sampling.

5 Experiments

For testing the general performance of the CSE algorithm and for comparing the different levellings presented above we run experiments on the Wang and Hu data set that was used in [14].

5.1 Building the Test Set

The Wang and Hu set consists of HTML-documents in which all target tables are marked using a special boolean attribute in the table node. This makes the set a ready-to-use ground truth for the table location task. Since CSE tries to solve the table recognition task as well, we have to extend the ground truth provided by Wang and Hu by additionally solving the recognition of table cores.

We decided to do this automatically with another table extractor, named *RE extractor* or *REE* in short, that uses regular expressions based on the relevant element names of HTML. Attempts to use the parsing of the document tree failed for too many documents where we tried three freely available parsers including the lxml-library [9].

REE uses a pattern for target tables based on the element name *table* and the additional attribute in order to get the substrings that represent the content of a target table. To solve the recognition task it applies an entry pattern (element name *td*) on the content of the matches of the row pattern (element name *tr*), which in turn is applied on the substrings matching the target table pattern. Matches of the row pattern that contain matches for headers (element name *th*) are ignored.

REEs capability for solving the table recognition task is limited. One minor problem is broken HTML in the inspected substrings. The main issue is meta data that is not declared as such. Still, we belief that the output of REE is sufficient for our experiments since extraction capability of CSE is low anyway.

In the following we specify two versions of the ground truth. Both are based on the output of REE but they apply filters of different strength. While for the *full set* only weak filtering is applied, *feasible set* contains those cases only that fulfill the assumption made by CSE.

Full Set

The Wang and Hu data set contains a total of 1393 documents. For the *full set* we only include the examples that contain at least one target table. CSE gets the number of tables k as promise and therefor has nothing to do if the promise is 0. Further, we bound the segmentation length to be not greater than 900. Documents with $p > 900$ are rare but they take a rather big fraction of the total runtime.

As a third filter criterion we demand that the extraction by REE is *successful* in the following sense: each extracted table should contain at least one row and any extracted row should have at least one entry. We hope that most cases where the table recognition by REE does not work as intended are detected by this criterion, while not to many table that are extracted as intended by REE fail to fulfill it. Table 1 shows the number of examples passing the filters discussed so far plus an additional filter discussed below.

Feasible Set

The feasible set is restricted to those documents from the full set in which all tables provided by the RE extractor fulfill the assumptions given in section 3.2. Though CSE may extract some of the table or part of tables from a document not in the feasible set, it is not possible that its output is entirely correct. The feasible is useful to analyze the discriminative power of in-column similarities as used by CSE and variants of the algorithm.

In order to fulfill assumptions a table has to be rectangular and it has to *fit in the content sequence* of the document. The latter means that the sequence of the entries in the table is a consecutive subsequence of the content part G^{e} of the alternating segmentation modulo the mapping γ. The conjunction of this two criteria is necessary for the assumptions to hold, but unfortunately it is not sufficient because of the implicit column headers. However, this problem can only be solved by human inspection and we therefor prefer to the use the above criteria as approximation. The number of documents under *hold assumptions* in table 1 refers to this approximation.

Although the problem of implicit column headers remains, the fraction of documents from which REE erroneously extract meta data should be lower in the feasible set because the rectangularity condition filters out cases with implicit titles or separating rows that are given as rows with one or no entry.

5.2 Performance Measure

For the evaluations of an extractors we need a loss function L^{y} that compares its outputs to the output provided as ground truth and encodes the comparison as a value in $[0, 1]$. The definition of the loss goes along the hierarchical structure of the outputs them self: L^{y} is an extension of a loss on tables L^{t} that is an extension of a loss on rows L^{r} which in turn is an extension of a loss on entries L^{e}.

We say that an extension is *strict* if the resulting loss is 1 whenever the two arguments do not have the same number of components and otherwise is given as aggregation of the *component losses* that are the losses on the pairs of components one from each of the two arguments having an identical index. The *average extension* is a strict extension which aggregate by the mean and the also strict *maximum extension* takes the maximum as aggregation. For instance, we obtain the 01 *loss* that checks whether its arguments are equal down to their entries by applying the maximum extension at any level of the hierarchy. Here, we want use a loss on table list which is more soft to several regards as we define in the following in a bottom-up manner.

The loss L^{e} on two entries is the 01 loss L^{s} on strings applied to the entries reduced by the peeling function γ introduced in section 3.2.

$$L^{\mathrm{e}}\left(e, \bar{e}\right) = L^{\mathrm{s}}\left(\gamma(e), \gamma(\bar{e})\right) \tag{18}$$

While the row loss L^{r} is given as strict maximum extension of the loss on entries, we want to use a soft table loss L^{t} such that dropping rows at the beginning or the end of a table results only in a gradual increase of loss.

Therefor, we define the table loss L^{t} not as a strict extension but as *best overlap extension* of the row loss. This extension searches an optimal way to match consecutive subsequence of component indexes of the argument with the smaller number of components to the longer one. For every component of the longer argument that is not matched a loss of 1 is taken into account.

Let $t = (r^1, \ldots, r^n)$ and $\bar{t} = (\bar{r}^1, \ldots, \bar{r}^{\bar{n}})$ be two tables that are to be compared where we assume without loss of generality that $n \leq \bar{n}$. In order to simplify the below definition of the loss L^{t} on the tables, we extend the shorter table t to $\tilde{t} = (\tilde{r}^1, \ldots, \tilde{r}^{\bar{n}+n+\bar{n}})$ by adding $\bar{n}$ *false rows* to either end of t. A false row is a row r such that $L^{\mathrm{r}}(r, \bar{r}) = 1$ for any row $\bar{r}$.

$$L^{\mathrm{t}}\left(t, \bar{t}\right) = \min_{d=0,\ldots,n+\bar{n}} \frac{1}{\bar{n}} \sum_{i=1}^{\bar{n}} L^{\mathrm{r}}\left(\tilde{r}^{d+i}, \bar{r}^{i}\right). \tag{19}$$

The minimization is needed because we want to define a loss depending only on the outputs decoupled from the input as pointed out in 3.1. If we toke the entry slices on the input string into account, we could use them to match the rows directly.

total	$k > 0$	$p \leq 900$	RE successful	hold assumptions
1393	774	727	700	162

Table 1: The number of examples in the Wang and Hu set that passed the filters in conjunction from the left to right up to the filter heading the column.

Finally, the table loss is expanded to a loss on table lists L^y by applying the average extension. The strictness of the extension is not an issue here because the CSE extractor uses promises on the number of tables. We refer to output loss L^y as *best row overlap loss* or *BRO loss* in short.

5.3 Results

We evaluated the performance of the CSE extractor by comparing its output to the REE-ground-truth discussed in subsection 5.1 in terms of best row overlap loss defined subsection 5.2. The CSE extractor gets the number of tables obtained from REE as promise. To simplify the implementation we let CSE look only for tables with $n \leq 80$, $m \leq 20$, other candidate tables were ignored.

In section 4.4 we pointed out that one has to adjust the scores depending on the shape of the candidate table in order to make the extraction by average entry similarities work. The discussed levellings try to do this adjustment in a specific, roughly motivated way. Alternatively, one may pass on the motivation and try to make a good guess on a function of m and n and use this as *ad hoc levelling*. For instance, one might multiply the scores by

$$\frac{1}{c(m,n)} = \gamma + n^{\beta} m^{\alpha\beta} \quad (20)$$

for some α, β and γ. Of course, we do not know a priory how to these set parameters. One can fit them using a training set, but we not try this possibility for this work. Still, we used the above class of ad hoc levellings for two purposes. First, by varying the parameters we get a rough impression on the impact of changes in the levelling. Second, we consider it as base line in the sense that the results yielded by levelling from section 4.4 should be comparable to this ad hoc levelling at least if the chosen parameters had not seriously been fitted to the test set. As a matter of fact, it was not difficult to find parameters for the ad hoc levelling in (20) that give better result than the more elaborated levellings discussed in section 4.4.

In general, the extraction performance of CSE with features from section is rather poor: the BRO losses are 0.825 and 0.613 for the full set and feasible set respectively using ad hoc levelling with $\alpha = \beta = \gamma = 0.5$. Results on the feasible set for fair and variance levelling based on Bernoulli sampling with different success probabilities q, for fair levelling with shuffled sampling, as well as for one ad hoc levelling are given in table 2.

The Parameters for the levellings in table 2 were chosen as follows. For the ad hoc levelling the result refers to the optimal values where we tested all combination with $\alpha = 0.2, 0.4, 0.6, 0, 08$, $\beta = 0.2, 0.3, \ldots, 1.4$ and $\gamma = 0, 1, 2, 8, 32$. The success probability for the iid Bernoulli segment sampling was tested at $q = 0.1, 0, 2, 0.3, 0.4, 0.5$ and the values that yielded the best and the worst performance respectively are in given in the table. Therefor, neither of those performances can be stated as performance for respective type of levelling as the parameters are fitted to the test set. The fair levelling with shuffled sampling is based on samples that are also taken from the Wang and Hu set but do not belong to the test set as they do not contain tables.

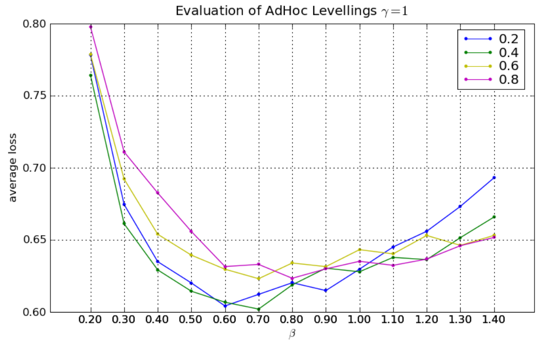

Figure 1: The mean BRO loss obtained by CSE with ad hoc levelling for different values of α and β measured on the feasible set. Different values of α are plotted with different colors while β runs along the x-axis.

Figure 1 shows the BRO losses for ad hoc levelling with $\gamma = 1$. The corresponding plots for other values of γ have similar shapes where with increasing γ the low loss region shifts towards greater values of β and the graphs for different α approaches to each other.

6 Conclusion

We investigated in a simple approach to the task of table extraction: first, we reduce the output space such that we can afford to inspect any candidate output, then, we select a given number of candidates with high average in-column similarities. The inspection of a set of HTML documents revealed that the proposed reduction of the output space cannot be applied in too many cases. Experiments on the remaining cases gave a first impression on the discriminative power of in-column similarities: even with more elaborated entry similarities than the simple ones applied here, it is presumably to weak to sufficiently solve the extraction task. On the other hand, given the simplicity of the applied similarities, we find that the extraction performance is on a level that justifies deeper investigation how we can effectively use the information that lies in similarities of column entries. In the following we revisit some issues of the this approach and indicate proposals for future work.

6.1 Alternating Segmentations

In section 3.2 we defined the input segmentation in terms of SGML tags but the concept of alternating segmentation is more general. Instead of tag-segments vs non-tag-segments on texts with markups, one might consider other input types with other alternating segmentations. A sufficient condition for an alternating segmentation in general terms is that no suffix of a separator-segment is a prefix of a content-segment *or* vice versa. Further, we can build different alternating segmentations and jointly maximize over the contained candidate tables, provided that the scores functions yield comparable values.

Ad Hoc	Fair Bern. $q = 0.5$	Fair Bern. $q = 0.2$	Fair Shuffled	Var. Bern. $q = 0.1$	Var. Bern. $q = 0.4$
0.598	0.655	0.617	0.623	0.664	0.628

Table 2: BRO losses of CSE using different type of levelling measured on the feasible set. The parameters for the ad hoc levelling in the first column are $\alpha = 0.4, \beta = 0.6$ and $\gamma = 0$ yielding the lowest among all tested combinations. For fair and variance levelling the two given values of q yielded the worst and the best result among five tested values from 0.1 to 0.5.

6.2 Restrictive Assumptions

The assumptions formulated in section 3.2 are too restrictive. Only one out four documents from the full set does fulfill them. Partially this caused by meta data rows in the ground truth provided by REE but we believe that that fraction would not increase to reasonable level even if we cleaned the example output by hand. It should be noted that most relaxations cause a exponential blowup of the search space. For instance, if we allow table rows to be separated by one or two separator segments instead of exactly one, the number of candidate table starting at given position grows exponentially in the number of rows at least as long as there are segments left in the segmentation. It is not obvious how to solve the maximization efficiently under such a relaxation. We cannot apply standard dynamic programming along the sequence of segments, because the column scores as given in section 4 does not factorize along this sequence.

6.3 Evaluation of Levellings

Except for levelling with shuffled sampling, all levellings that have been applied in our experiments are parametrized. As long as we do not provide algorithms for pre-test determination of the parameters a comparison of levellings schemes based on the obtained performances is delicate. But we might say that fair and variance levelling as proposed in section 4.4 do not provide a adequate technique for boosting the performance of CSE compared to ad hoc levelling since competitive parameters can easily be found for the later. The proximate way to make comparison between parametrized levellings is to fit each of the levellings with respect to a training set. This will also give us an idea to what extend the performances differ on disjoint test sets which is important to know when we decide on the effort to put in the selection of the levelling.

6.4 Training of Similarity Functions

The definition of the kernels given in sections 4.3 were made ad hoc. We belief that more elaborated similarities can improve the performance of the CSE algorithm. Instead of building improved similarities by hand, we may adapt the kernels with respect to the extraction task using training examples. The table score H is linear in any linear parameterization of the similarities, for instance linear combinations of fixed kernels. Provided such a parametrization, we can apply generic training schemes for linear models as the one proposed in [12]. However, for sake of linearity, we have to restrict ourself to documents that contain one table only. Further, we depend on a fast convergence since the evaluation of H is expensive.

References

[1] J. L. Balcazar, J. Diaz, and J. Gabarro. *Structural Complexity I*. EATCS Monographs on Theoretical Computer Science. Springer Verlag, 1993.

[2] H.-H. Chen, S.-C. Tsai, and J.-H. Tsai. Mining tables from large scale HTML texts. In *International Conference on Computational Linguistics (COLING)*, pages 166–172. Morgan Kaufmann, 2000.

[3] W. Gatterbauer and P. Bohunsky. Table extraction using spatial reasoning on the CSS2 visual box model. In *National Conference on Artificial Intelligence (AAAI)*. AAAI Press, 2006.

[4] W. Gatterbauer, P. Bohunsky, M. Herzog, B. Krüpl, and B. Pollak. Towards domain-independent information extraction from web tables. In C. L. Williamson, M. E. Zurko, P. F. Patel-Schneider, and P. J. Shenoy, editors, *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 71–80. ACM, 2007.

[5] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. J. Smola. A kernel method for the two-sample-problem. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Neural Information Processing Systems*, pages 513–520. MIT Press, 2006.

[6] M. Hurst. Layout and language: Challenges for table understanding on the web. In *In Web Document Analysis, Proceedings of the 1st International Workshop on Web Document Analysis*, pages 27–30, 2001.

[7] N. Kushmerick. Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence*, 118(1–2):15–68, 2000.

[8] A. Lee. *U-Statistics: Theory and Applications*. Marcel Dekker Inc., New York, 1990.

[9] lxml: pythonic binding for the libxml2 and libxslt libraries. http://codespeak.net/lxml/index.html.

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

[11] A. Pivk. Automatic ontology generation from web tabular structures. *AI Communications*, 19(1):83–85, 2006.

[12] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 104, New York, NY, USA, 2004. ACM Press.

[13] M. J. Wainwright and M. I. Jordan. Semidefinite relaxations for approximate inference on graphs with cycles. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *Neural Information Processing Systems*. MIT Press, 2004. (long version).

[14] Y. Wang and J. Hu. A machine learning based approach for table detection on the web. In *Proceedings of the Eleventh International World Wide Web Conference*, pages 242–250, 2002.