5-2017

Improving Automatic Content Type Identification from a Data Set

Kathy T. Dai
University of Arkansas, Fayetteville

IMPROVING AUTOMATIC CONTENT TYPE IDENTIFICATION FROM A DATA SET

Kathy Dai

Bachelor of Science in Computer Science
University of Arkansas, Fayetteville
May 2017

1.0 INTRODUCTION

Big Data refers to the massive amount of data in the digital world today that is generated from many different methods such as medical records, applications, and largely the internet [1]. These data sets are gathered every day in large volumes. By capturing all the data from customers, businesses can develop strategies and products based on their user demographic [2]. *Big Data* in the technological sense refers to the numerous programs that can quickly and efficiently organize and sort this data so that businesses can understand and appeal to their customers, thereby improving sales and revenue [3].

Content-based oracles are a *Big Data* technique that deal with specific types of data, flat files consisting of personal records with a fixed number of fields or variable length fields, such as the below Figure 1.

Kenneth Mitchell	5547 East Eighth Court	Apt. 70	KITTANNING	PA

Figure 1: Fully-fixed record from Small.txt

Traditionally, processing these records require manual labor, in the sense that the user would open the file and determine which field is the first name, middle name, and etcetera. This process is inefficient, time-consuming and prone to human error. Content-based oracles attempt to automate the process of "inferring the specified meta data" [4], [5]. This thesis researches the expansion on these content-based oracles that were developed as part of Dr. Reid Phillips' PHD dissertation and how to improve and document these content-based oracles for higher accuracy.

Throughout the majority of this research, we will be dealing with content-type identification to improve the file schema inference using Dr. Reid Phillips' PHD dissertation [6]. The tools provided are the content-based oracles [7]. There are a total of 20 oracles, one for each type of metadata (first name, middle name, and etcetera). Each oracle acts as a database. When all the oracles are ran against a field, we mean that the field is compared to each oracle' database in search of hits. Because the name of the oracle corresponds to the database it contains, we are able to use the terms "content type" and "oracle" interchangeably.

2.0 BACKGROUND

2.1 Related Work

In a different perspective, content-based oracles can be seen as machine-learning. Blaylock and Allen researches "a system which uses statistical, corpus-based machine learning techniques to perform instantiated goal recognition – recognition of both a goal schema and its parameter values" using a "parameter recognizer" [8]. These are more advanced due to their capability to recognize new values to be under unsupervised learning allowing for dynamic databases. Unsupervised learning is the ability to determine outputs for a collection of inputs without having direct access to the correct output [9]. The content-based oracles used in this research use static databases that require manual updates instead of artificial intelligence.

2.2 Goals and Purpose

The automation of processing this information is dependent on programming logical decisions based on human analysis. For example, when looking at Figure 1, we must consider these questions to correctly identify the field for "Kenneth": Is this a name? Is this a street name? Is this a city?

Suppose the human decides that "Kenneth" is a name. We must further break it down by asking different questions: Is this a first, middle, or last name? Suppose the human decides that

"Kenneth" is a first name. These content-based oracles are attempting to simulate this logic based on numerical values. How does the human decide in the beginning that "Kenneth" is a name? It is more likely to be a name than a street name or a city, resulting in a hit system. The content-based oracles go through many records like Figure 1 and record all the hits. In this case of "Kenneth", the name, street name, and city oracles should be recorded with hits because that is what a human considered. In the content-based oracles, a hit is considered if the field matches an entry within its corresponding lookup table. For the example, "Kenneth" would be found as an entry within the name, street name, and city lookup tables.

2.3 The Prototype Program

The program in the research takes an input file that contains *n* amount of personal information records and a configuration file. The input file can be of any type of encoding such as ASCII or EBCDIC [10], and therefore must be identified. Currently, there is a basic algorithm in place to determine the encoding between ASCII and EBCDIC [11].

At first glance, the layout of the file is unknown. The program must determine the base layout out of 3 types: fully fixed, delimited, and hybrid. The next step is to determine the record length. In order to discover the record length, each file type has different components that need to be determined [12].

In a fully fixed file, the start and end positions of the fields and records must be found. The program implements a trial and error algorithm and attempts all possible fields. With fully fixed files, the program also assumes that all the records in the file are of the same structure. The program then records in a line from the file and attempts different positions. For example, from column 1 to column 2 could be a name prefix. Column 1 to column 3 could also be a name prefix. As long as the number of hits within a range of columns is within some confidence interval of the maximum valid count, the program continues to increase the range. The logic behind is the assumption that as the length of the examined field increase, the number of recognize entries will increase. Eventually, the number of hits will hit a maximum limit or "plateau" and then suddenly decrease. At this point, we can move on to the next field. Whenever an initial record structure is determined, the program can then determine the record length. Then, the program checks with the other records in the file given the record length and determines whether the start and end positions with the first record "lines up" with the other records. If it does "line up", then the second record and the records after will repeat the initial record structure [13]. At this point, the fully fixed file is ready for content type identification.

In a fully delimited file, the delimiter must be identified. These can be commas, tabs, or any field separator character [14]. Once the delimiter is identified, the program learns how many fields are with a record, an average record length, and how many records are within the file. The file is now ready for content type identification.

In a hybrid file, the program must determine the record delimiter and the start and end positions of the field. The hybrid file uses fully delimited method to determine the record delimiter and the fully fixed method to determine the location of the fields. The hybrid file is now ready for content type identification.

2.4 Types of Files

This research deals with 3 total files: fully fixed, fully delimited, and hybrid. Fully fixed files contain fixed sized records. Each record is exactly a certain length. In addition, the oracles must determine the start and end position of each field before identifying the content type. When

the start and end positions are discovered, the fields are left to be determined. At this point, the fully fixed file acts as a delimited file with the start and end positions acting as delimiters. In a fully delimited file, each field is separated with a delimiter, and the records are delimited as well, such as a carriage return. But at times, files can have a carriage return as the record delimiter, but fixed length fields. This file is known as a hybrid file: a combination of both hybrid and fully fixed files. For the purposes of this research, fully fixed and hybrid files are considered to be in the same category.

2.5 Sampling

In *Big Data*, files can contain millions of records. To be able to infer the layout using every record can be time-consuming and costly. For example, if there are 1 million records with a total of 15 fields. Each field must be ran against each of the 21 oracles. For this math, we can say that it takes 1 millisecond to run 1 oracle, against 1 field, in 1 record.

$$1{,}000{,}000\ records \times 15\ fields \times 21\ oracles = 315{,}000{,}000\ ms$$
$$= 315{,}000\ \text{sec} = 5250\ minutes = 87.5\ hours$$

The idea of sampling is important because it essentially reduces 100 hours to a reasonable amount of time, while making an assumption: all the records in the file assume the same structure. In that case, if a record is laid out like First Name, Middle Name, Last Name, Address Line Two, Address Line One, then all the records take on this structure. In this research, a file with 42 records has 33 records sampled. A file with 2000 records have 72 records sampled.

However, the disadvantages of a sample is the possibility the oracles sample only records without a last name field. But because samples are usually random, it is unlikely that 72 records of 2000 records would not have a missing field, such as last name.

2.6 Conflict Resolution for Fully Delimited Files

The goal of conflict resolution is to find the best content-type for a given field. The process is similar to a tournament of conflicting oracles. In order to be eligible to enter the tournament, it must meet a minimum threshold and it must have some degree of certainty. After that, the conflicting oracles are sent into the tournament pipeline. The oracles with the highest rank goes on to the next round. For the final elimination, the conflicting oracles compare primary and secondary hits.

2.6.1 Minimum Threshold

For oracles to essentially become eligible "candidates", they must pass certain stipulations. Otherwise, any oracles that contain 1 or more hits would be fed into the pipeline, which would be incredibly inefficient. We must eliminate unlikely oracles instead. The first stipulation is to have a certain number of hits above a minimum threshold. This threshold is predetermined for all oracles and can be seen in Appendix A. By taking the current hits of the corresponding oracle (compared to a database of entries), it is divided by the number of records sample minus the blank count of that corresponding oracle, as shown in Equation 1 [15]. If that value is greater than the minimum threshold, then it is stored for later conflict resolution.

$$\frac{Hits_i}{RecordsSampled - Blank_i} > minimum\ Threshold$$

Equation 1: Minimum threshold equation as defined by the oracles.
i = corresponding oracle identifier number

A good example would be the name prefix field against the city oracle. We can assume that out of 100 records sampled and a blank count of 50, there are 10 hits for the name prefix field within the city oracle. This means that 50 records out of the 100 records sampled had blank records for the name prefix field. Out of the other 50, 10 records had a prefix entry that matched within the city oracle, resulting in a threshold of 20%. If we compared it to the city oracle's minimum threshold of 60%, we see that 20% did not reach the desired threshold, discarding it from the potential content types.

2.6.2 Uncertainty

After all the oracles that pass the minimum threshold are stored, these oracles are now considered the conflicting oracles. At this point, the oracles enter the decision process that chooses the best content type for the field. Before each oracle is compared against one another, the conflicting oracles undergo another stipulation. This second stipulation is whether the conflicting oracle is certain. The definition of uncertain is when the value of the corresponding blank count over the number of records sampled is greater than or equal to the corresponding max blank percentage (Appendix A). Equation 2 displays this in equation form. The purpose of this uncertainty check is to consider only the conflicting oracles that have more hits than blank counts because this would mean they are more likely to be the better fit.

$$\frac{Blank_o}{Records} \geq MaxBlank_o$$

Equation 2: Uncertainty equation as defined by the oracles.

Assume that the name prefix field is being compared against the name prefix oracle. Also, assume that 100 records are being sampled, and 10 of those records contain a blank entry for the name prefix field. This results in a 10% blank percentage. In name prefix, the maximum blank percentage is 98%. Because 10% is less than the maximum blank percentage, we can say that the name prefix oracle is certain and pass it on to the final decision.

2.6.3 Final Decision

After the uncertainty check, we enter the decision phase where two conflicting oracles are compared against another and the "winner" goes on to be compared to the next conflicting oracle.

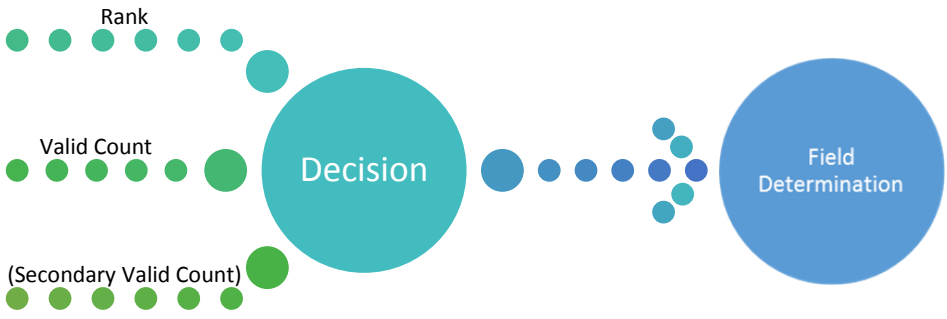


Figure 2: Oracle decision logic

Rank is the first factor considered in the decision process. The higher the rank of a given oracle, the more likely that specific oracle will be chosen as the field. The ranks are pre-defined

values of the oracles, demonstrating reliability. An oracle with the rank of 750 is a more reliable oracle than one with a rank of 50. For example, take street name as the oracle of rank 750 and directional (North, South, etc.) of rank 50. The oracles deem it more reliable to have a street name oracle than a directional oracle because not all addresses have a directional but most addresses have street names. Then, the decision takes into consideration the valid (primary) count, or the number of hits for that oracle. If there is a secondary valid count, then that has higher weight than the valid count and has the "final" say in the decision towards field determination.

The secondary valid count is an optional value given to certain oracles. The purpose of the secondary valid count is to check which of the primary count hits are still valid. If we take the last name oracle for example, some entries that could be considered primary hits are "Adam", "Johnson" and "White". Out of these 3 values, "Adam" is likely to be first names than last names, but "Johnson" and "White" are more likely to be last names. For this instance, the secondary count would be 2. These oracles would have reduced reference tables that are subsets to their original database. The reason behind the secondary count being used over the primary count (when available) is because it allegedly is a more accurate reading of the hits.

For example, consider the first name and last name oracle. After the primary hits and secondary hits have been calculated, let's say first name has a primary count of 100 and a secondary count of 60. Last name has a primary count of 110 and a secondary count of 50. Because these two oracles contain a secondary count, the secondary count is used within Equation 3 over the primary hit to determine the best content type for the current unknown field.

$$\frac{Hits}{RecordsSampled - BlankCount} = percentValid$$

Equation 3: Percent valid equation for both primary and secondary counts. Hits can be the primary count or the secondary count depending on if there is a secondary count available.

Within the existing code for the content-based oracles, there are 7 oracles displayed in Figure 5 that would have this optional secondary value count. These oracles have additional, more concise reference tables.

	Oracle
1	First Name
2	Last Name
3	Full Name*
4	Street Name
5	Address Line One*
6	Address Line*
7	Cities

Table 1: Oracles with secondary valid count
*: Not investigated

3.0 CONTRIBUTION

By using in-depth analysis, the existing content-based oracles undergo a debugging process to understand and document all the logic within the decision-making process for fully delimited files. Currently, the content-based oracles are not 100% accurate. Once it is fully understood, improvements can be made upon the knowledge to increase the accuracy.

4.0 EXPERIMENTATION

4.1 FullyParsedFixed – Given file

To obtain an idea of how accurate the oracles were, I ran all the files against the oracles and studied the results for each input file. I began with the file FullyParsedFixed, a file provided within the source code because at first glance, it seemed to have the most issues. Within FullyParsedFixed (a hybrid file with a record delimiter, but fixed lengths for each field), there were mislabeled fields and 4 total unspecified fields. Figure 3 displays the layout structure built for FullyParsedFixed on the first run. FullyParsedFixed is considered a hybrid file with fixed length fields and a carriage return as a record delimiter. Table 2 displays the correct layout structure for FullyParsedFixed.

```
RECORDS EXAMINED: 150
RECORD LENGTH: 232
LAYOUT ---
        CONTENT TYPE NAME         J   SP   EP LEN       PC   SC   BC
    1.)   name prefix              L    1   10  10 --    44   -1   99
    2.)   first name               L   11   40  30 --   150  144    0
    3.)   middle name              L   41   70  30 --   122  122   28
    4.)   last name                L   71  100  30 --   150  139    0
    5.)   * UNSPECIFIED *          L  101  160  60 --     0   -1    0
    6.)   street suffix            L  161  167   7 --   129   -1    0
    7.)   last name                L  168  170   3 --    34    0   95
    8.)   * UNSPECIFIED *          L  171  180  10 --     0   -1    0
    9.)   zipcode                  L  181  185   5 --   150   -1    0
   10.)   city                     L  186  215  30 --   150  101    0
   11.)   state                    L  216  217   2 --   150   -1    0
   12.)   * UNSPECIFIED *          L  218  222   5 --     0   -1    0
   13.)   last name                L  223  225   3 --     4    0  145
   14.)   * UNSPECIFIED *          L  226  232   7 --     0   -1    0
----------
```

Figure 3: Layout structure for FullyParsedFixed.txt

Field Position	Column Name	Start Position	End Position
1	Name Prefix	1	10
2	First Name	11	40
3	Middle Name	41	70
4	Last Name	71	100
5	Name Suffix	101	110
6	Primary_Number	111	120
7	Pre_Directional	121	130
8	Street	131	160
9	Street_Suffix	161	170
10	Post_Directional	171	180
11	Zip Code	181	185
12	City	186	215
13	State	216	217
14	Unit_Designator	218	227
15	Secondary_Number	228	232

Table 2: Expected layout structure for FullyParsedFixed.txt

In Figure 3, we can see that fields 1, 2, 3, and 4 are identified correctly to name a few. Unit designator (field 13 in Figure 3) and secondary number (field 14 in Figure 3) seemed to be mislabeled or unspecified fields. When looking at the resulted field location with the actual field location in Table 2, it is clear that determining the field location can be improved. This is most likely the reason why there are errors with the results of this file. The second reason could be within the content type oracles.

4.1.1 FullyParsedFixed - Delimited

In the Figure 3, we can see field 8 has the correct start and end positions, but an UNSPECIFIED content type. This was the post-directional field. The directional oracle handles all cardinal directions, so it was peculiar to see that the field was unspecified. To investigate this further, we need to understand the root of the problem: determining the field location (start and end positions) or the content-type identification. I hypothesized that it was determining the field location that might be interfering with the content type identified because of my observations for Figure 3 and Table 2. To eliminate the field location, I transformed this fixed file to a fully delimited by inserting delimiters at the end positions. The results are shown in Figure 4. By doing so, we eliminate the oracles' job of determining the field location, since the fields are provided between delimiters. If it is able to identify the content type with delimiters, than the issue is with the combinatorics approach of determining the field length.

```
RECORDS EXAMINED: 150
RECORD LENGTH: 15
LAYOUT ---
       CONTENT TYPE NAME           SP       PC   SC   BC
   1.)   name prefix                1 --    44   -1   99
   2.)   first name                 2 --   150  144    0
   3.)   street name                3 --   122   34   28
   4.)   last name                  4 --   150  139    0
   5.)   * UNSPECIFIED *           5 --     0   -1    0
   6.)   street number              6 --   150   -1    0
   7.)   directional                7 --    49   -1  101
   8.)   city                       8 --   114   98    0
   9.)   street suffix              9 --   150   -1    0
  10.)   directional               10 --    50   -1  100
  11.)   zipcode                   11 --   150   -1    0
  12.)   city                      12 --   150  101    0
  13.)   state                     13 --   150   -1    0
  14.)   * UNSPECIFIED *          14 --     0   -1    0
  15.)   * UNSPECIFIED *          15 --     0   -1    0
----------
```

Figure 4: Results from the delimited FullyParsedFixed.txt

In Figure 4, we see that fields 1, 2, and 4 are identified correctly, but field 3 is mislabeled street name, and field 5 in unspecified. However, overall, the delimited file was able to find all 15 fields, while the hybrid file was only able to find 14 fields (Figure 3), with some questionable fields. Looking at these results, it seems that my hypothesis was correct, revealing an issue with the field location than the content-type identification. Using these results, I wanted to replicate the experiment on a smaller scale, with less variation. We come back to these results in Section 4.3, because by transforming FullyParsedFixed into a delimited file, we essentially use FullyParsedDelimited, the delimited version of FullyParsedFixed with no trailing space.

4.2 SmallParsed – Created file

Using a similar record structure to FullyParsedFixed in Table 3, I created a file containing 42 records like Figure 5, but instead of fixed lengths for the field, I decided to

simplify the file to a fully delimited type. In addition, these records had entries that were identical to each other in fields. In other words, for all the records, the middle name field contained "Ethan". When passing the SmallParsed.txt to the oracles, I yielded the results in Figure 6.

Mother,Adam,Ethan,Young,PHD,567,NW,2nd,Wall,NW,80645,La Salle,AL,Unit,R

Figure 5: Example record of SmallParsed.txt

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Name Prefix*	First Name	Middle Name*	Last Name	Name Suffix*	Street Number	Pre-Directional	Street Name	Street Suffix	Post-Directional*	Zip code*	City*	State	Unit Designator*	Unit Number*

Table 3: Record structure in SmallParsed.txt
*: Fields remained the same for all records

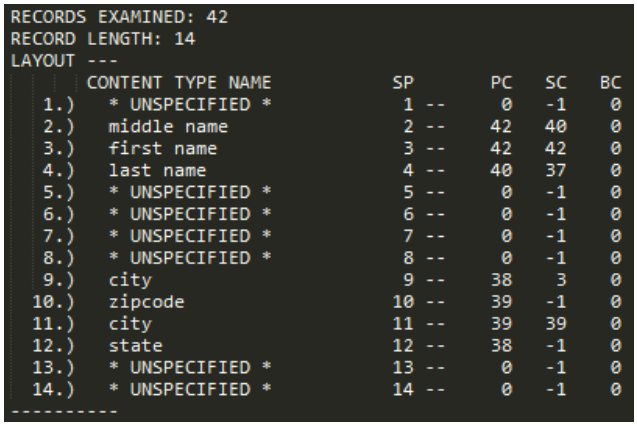
```
RECORDS EXAMINED: 42
RECORD LENGTH: 15
LAYOUT ---
      CONTENT TYPE NAME          SP       PC   SC   BC
  1.)   * UNSPECIFIED *          1 --     0   -1    0
  2.)   middle name              2 --    42   40    0
  3.)   first name               3 --    42   42    0
  4.)   last name                4 --    40   37    0
  5.)   * UNSPECIFIED *          5 --     0   -1    0
  6.)   * UNSPECIFIED *          6 --     0   -1    0
  7.)   * UNSPECIFIED *          7 --     0   -1    0
  8.)   * UNSPECIFIED *          8 --     0   -1    0
  9.)   city                     9 --    38    3    0
 10.)   * UNSPECIFIED *         10 --     0   -1    0
 11.)   zipcode                 11 --    39   -1    0
 12.)   city                    12 --    39   39    0
 13.)   state                   13 --    38   -1    0
 14.)   * UNSPECIFIED *         14 --     0   -1    0
 15.)   * UNSPECIFIED *         15 --     0   -1    0
----------
```

Figure 6: Results of SmallParsed.txt with base configuration

In Figure 6, I ran SmallParsed.txt through the oracles to obtain the base results to be used as comparison with various changes. The oracles were unable to label the name prefix, name suffix, pre-directional, and street number, and street name, but did identify fields 12 and 13 correctly.

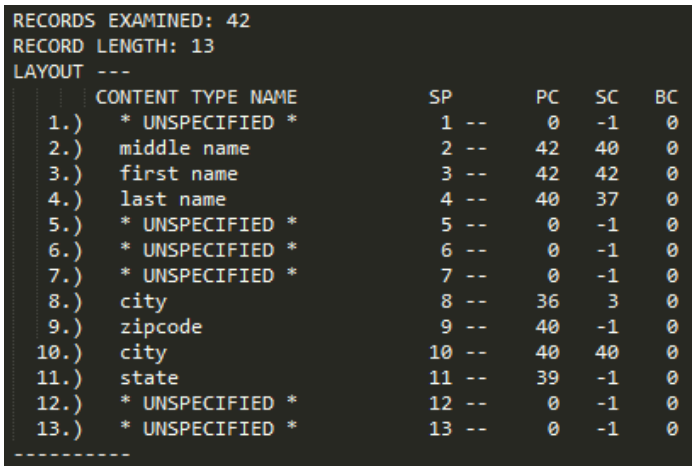
4.2.1 - Directionals

For this next part, I focused on the directionals within the record because of the phenomena observed in Figure 3 (the exact field location was correct, but the content type was unspecified). I sought to reduce Address Line One, removing the post-directional field from all the records (Figure 7), then eventually removing the pre-directional as well (Figure 8).



```
RECORDS EXAMINED: 42
RECORD LENGTH: 14
LAYOUT ---
      CONTENT TYPE NAME          SP       PC   SC   BC
   1.)   * UNSPECIFIED *          1 --     0   -1    0
   2.)   middle name              2 --    42   40    0
   3.)   first name               3 --    42   42    0
   4.)   last name                4 --    40   37    0
   5.)   * UNSPECIFIED *          5 --     0   -1    0
   6.)   * UNSPECIFIED *          6 --     0   -1    0
   7.)   * UNSPECIFIED *          7 --     0   -1    0
   8.)   * UNSPECIFIED *          8 --     0   -1    0
   9.)   city                     9 --    38    3    0
  10.)   zipcode                 10 --    39   -1    0
  11.)   city                    11 --    39   39    0
  12.)   state                   12 --    38   -1    0
  13.)   * UNSPECIFIED *         13 --     0   -1    0
  14.)   * UNSPECIFIED *         14 --     0   -1    0
----------
```

Figure 7: Results of SmallParsed.txt without Post-Directionals



```
RECORDS EXAMINED: 42
RECORD LENGTH: 13
LAYOUT ---
      CONTENT TYPE NAME          SP       PC   SC   BC
   1.)   * UNSPECIFIED *          1 --     0   -1    0
   2.)   middle name              2 --    42   40    0
   3.)   first name               3 --    42   42    0
   4.)   last name                4 --    40   37    0
   5.)   * UNSPECIFIED *          5 --     0   -1    0
   6.)   * UNSPECIFIED *          6 --     0   -1    0
   7.)   * UNSPECIFIED *          7 --     0   -1    0
   8.)   city                     8 --    36    3    0
   9.)   zipcode                  9 --    40   -1    0
  10.)   city                    10 --    40   40    0
  11.)   state                   11 --    39   -1    0
  12.)   * UNSPECIFIED *         12 --     0   -1    0
  13.)   * UNSPECIFIED *         13 --     0   -1    0
----------
```

Figure 8: Results of SmallParsed.txt without Pre-Directionals

Because removing the post-directional field didn't yield any improvement, I proceeded to further reduce the field by removing the pre-directionals as well. However, doing so still did not reveal any change, so instead of field reduction, I attempted field variation (Appendix D). Perhaps, there was some mechanism that looked at all the records and labeled them unspecified if all the fields had the same entry. The oracles might be able to recognize the fields with other hits for different entries than with hits for the same entry. My hypothesis was incorrect when the results shown in Figure 9 yielded no change.

```
RECORDS EXAMINED: 42
RECORD LENGTH: 14
LAYOUT ---
      CONTENT TYPE NAME          SP       PC   SC   BC
   1.)   * UNSPECIFIED *          1 --     0   -1    0
   2.)   middle name              2 --    42   40    0
   3.)   first name               3 --    42   42    0
   4.)   last name                4 --    40   37    0
   5.)   * UNSPECIFIED *          5 --     0   -1    0
   6.)   * UNSPECIFIED *          6 --     0   -1    0
   7.)   * UNSPECIFIED *          7 --     0   -1    0
   8.)   * UNSPECIFIED *          8 --     0   -1    0
   9.)   city                     9 --    38    3    0
  10.)   zipcode                 10 --    39   -1    0
  11.)   city                    11 --    39   39    0
  12.)   state                   12 --    38   -1    0
  13.)   * UNSPECIFIED *         13 --     0   -1    0
  14.)   * UNSPECIFIED *         14 --     0   -1    0
----------
```

Figure 9: Results of SmallParsed.txt with varying Pre-Directionals

I decided that the next step was looking into each individual oracle and understand the decision logic behind the unspecified and the mislabeled fields. I first began with field 9 (Figure 9), which is supposed to be the street suffix field, but was labeled as a city field.

4.2.2 – Street Suffix

Looking at the potential field locations from the results in Figure 9, we are able to see that street suffix and city were the two conflicting oracles for field 9. Street suffix had rank of 750, but city had a rank of 1000 (Appendix A), removing street suffix from the list of conflicting oracles. Understanding this, I looked at all the different data entries within SmallParsed, to see if any of the street suffixes could be hit as cities. After doing some research, I could not find a city called “street”, “road”, or “avenue”, and removed it from the city oracle’s maximum database (primary count is populated from the number of hits in the maximum database) and reran by experiment. As a side note, I decided to keep post-directionals removed from this run because the addition of it would not affect the results since it would only add an unspecified field to the results.

```
RECORDS EXAMINED: 42
RECORD LENGTH: 14
LAYOUT ---
      CONTENT TYPE NAME          SP       PC   SC   BC
   1.)   * UNSPECIFIED *          1 --     0   -1    0
   2.)   middle name              2 --    42   40    0
   3.)   first name               3 --    42   42    0
   4.)   last name                4 --    40   37    0
   5.)   * UNSPECIFIED *          5 --     0   -1    0
   6.)   street number            6 --    42   -1    0
   7.)   directional              7 --    42   -1    0
   8.)   * UNSPECIFIED *          8 --     0   -1    0
   9.)   street suffix            9 --    42   -1    0
  10.)   zipcode                 10 --    39   -1    0
  11.)   city                    11 --    39   39    0
  12.)   state                   12 --    38   -1    0
  13.)   * UNSPECIFIED *         13 --     0   -1    0
  14.)   * UNSPECIFIED *         14 --     0   -1    0
----------
```

Figure 10: Results of SmallParsed.txt with improved city oracle.

In Figure 10, we see that field 9 is correctly identified as street fix. In removing the 3 entries, we were able to decrease the primary hits for city. The city oracle then does not reach the minimum threshold and was not considered a conflicting oracle, leaving street suffix the only conflicting oracle in the decision pipeline, rendering it the victor. We also notice here that street number and directional are correctly identified, but looking at the potential field locations, there is no difference in primary count, secondary count, or blank counts between figure 9 and figure 10 (Appendix E). A possible speculation is that street number and directional were correctly identified because once street suffix was recognized, the oracles recognize all of these fields as part of address line one and was able to label the other two fields instead of unspecified.

4.3 FullyParsedDelimited – Given file

Next, the logic is to attempt this new city oracle on a different file where more records are sampled, to see if the logic is consistent. FullyParsedDelimited.txt is a generated file of information containing about 2000 records provided with the oracles' source code which allow us to have a more accurate and larger sample size for the oracles to make a better inference of the layout. Figure 12 displays the results of FullyParsedDelimited without the removal of entries of the city oracle and Figure 13 displays the results after the removal of entries.

```
RECORDS EXAMINED: 72
RECORD LENGTH: 15
LAYOUT ---
       CONTENT TYPE NAME          SP       PC   SC   BC
   1.)   name prefix              1 --    25   -1   43
   2.)   first name               2 --    72   69    0
   3.)   street name              3 --    61   21   11
   4.)   last name                4 --    72   69    0
   5.)   name suffix              5 --     9   -1   60
   6.)   * UNSPECIFIED *         6 --     0   -1    0
   7.)   * UNSPECIFIED *         7 --     0   -1    0
   8.)   city                     8 --    50   41    0
   9.)   city                     9 --    50    0    0
  10.)   * UNSPECIFIED *        10 --     0   -1    0
  11.)   zipcode                 11 --    72   -1    0
  12.)   city                    12 --    72   51    0
  13.)   state                   13 --    72   -1    0
  14.)   * UNSPECIFIED *        14 --     0   -1    0
  15.)   * UNSPECIFIED *        15 --     0   -1    0
----------
```

Figure 12: Results of FullyParsedDelimited.txt before oracle changes

```
RECORDS EXAMINED: 72
RECORD LENGTH: 15
LAYOUT ---
       CONTENT TYPE NAME          SP       PC   SC   BC
   1.)   name prefix              1 --    25   -1   43
   2.)   first name               2 --    72   69    0
   3.)   street name              3 --    61   21   11
   4.)   last name                4 --    72   69    0
   5.)   name suffix              5 --     9   -1   60
   6.)   street number            6 --    72   -1    0
   7.)   directional              7 --    24   -1   48
   8.)   city                     8 --    50   41    0
   9.)   street suffix            9 --    72   -1    0
  10.)   directional              10 --    21   -1   51
  11.)   zipcode                 11 --    72   -1    0
  12.)   city                    12 --    72   51    0
  13.)   state                   13 --    72   -1    0
  14.)   * UNSPECIFIED *        14 --     0   -1    0
  15.)   * UNSPECIFIED *        15 --     0   -1    0
----------
```

Figure 13: Results of FullyParsedDelimited.txt after oracle changes.

Field Position	Column Name
1	Name Prefix
2	First Name
3	Middle Name
4	Last Name
5	Name Suffix
6	Primary_Number
7	Pre_Directional
8	Street
9	Street_Suffix
10	Post_Directional
11	Zip Code
12	City
13	State
14	Unit_Designator
15	Secondary_Number

Table 4: Expected layout for FullyParsedDelimited.txt

Before the city oracle changes are applied, we see fields 1, 2, 4, 5, 11, 12, and 13 are identified correctly in Figure 12 using the reference table provided in Table 4. The results still revealed unspecified fields, but they also show fields 3 and 8 as mislabeled in Figure 13.

After the city oracle changes, we see improved accuracy. In Figure 13, we see that the goal in mind was to correctly identify street suffix instead of city for field 9. Like in SmallParsed.txt, we revealed fields 6, 7, and 10 in Figure 13 which were the 3 previously unspecified fields shown in Figure 12. Unfortunately, there are still a couple of issues. For example, the oracles have determined that the street name instead of the middle name is the best fit for field 3 in both Figure 12 and Figure 13. For field 8, city was determined the better fit instead of the correct field descriptor, street name.

Because the city oracles have increased the accuracy of the layout inference, we are hope to further increase accuracy by correcting the mislabeling of field 3 and field 8 as shown in both Figures 12 and Figure 13. By delving into the code, we discovered the logic described in Section 2.6.3 that determined the best fit field.

4.3.1 Middle Name

When investigating the issue of the middle name versus the street name (Figure 13, Table 4), I came to understand the logic that is indicated in Section 2.6 for conflict resolution. After the primary count and the secondary count for conflicting oracles are obtained, we are ready to decide. To assist in the elimination process, we check to see if the conflicting oracle is "uncertain." The principle of uncertainty is described in Section 2.6.2.

Some people do not have middle names. In another situation, most people live in a house that negates the need for a unit designator and unit number. However, most if not all people have a first name. Since these first name field is less likely to be blank, the corresponding oracles have a higher maximum blank count threshold. If the conflicting oracle is flagged uncertain, it is eliminated from the decision. When considering the other oracles that were considered for field 3 (Figure 13), I observed that first name and last name were flagged as

uncertain, eliminating middle name from the decision and leaving street name and city, where street name was determined the better fit.

The middle name oracle is an oracle whose results depend entirely on the name oracles, mostly the first name oracle. The middle name oracle specifies two instances that indicate a middle name: a single character such as an initial between name fields, or multiple first name fields. For the records sampled in FullyParsedDelimited, if first name was a conflicting oracle, the middle name oracle was also considered.

4.3.2 Street Name

For field 8 (Figure 13, Figure 14), I investigated the conflicting oracles for the street name: last name, city, and street name. Due to the decision logic explained in Section 2.5.3, we see that rank is encountered first for these 3 conflicting oracles. Because last name and city have a rank of 1000, and street name has a rank of 750, street name is discarded during the first phase of the decision logic, leaving city and last name to be the finalists in the decision, where city comes out on top due to its higher secondary count.

5.0 RESULTS

5.1 Small Synthetic File – SmallParsed.txt

Using this small file, we were able to improve the city oracle by removing 3 entries: "road", "street" and "avenue". These changes caused no negative impacts on other delimited files, but instead decreased the number of unspecified fields. In addition, we are able to provide more documentation for important code.

5.2 Large Synthetic File – FullyParsedDelimited.txt

Considering the city oracle changes, FullyParsedDelimited demonstrated increased accuracy compared to its previous test run without the changes. In addition to correcting the street suffix, it was able to correctly identify 3 unspecified fields. This research is also able to explain why middle name and street name are mislabeled.

6.0 FUTURE WORK AND CONCLUSIONS

We found room for improvement within the decision making logic within the content-based oracles, specifically due to the ranking system between street name and city discussed in Section 4.2.2. The reason being that even though street name and a higher number of valid hits, due to its rank being lower than city's rank ($750 < 1000$), it was eliminated from the decision process. However, I feel that this could be more accurate if a value was calculated; the oracle's rank is multiplied together with the primary count and the secondary count is added to the value. A value like this could be established and be a better indicator of determining the best fit. This method does not remove any layers of the decision logic, but instead, incorporates all the layers of the decision logic into one, so no oracle is unnecessarily eliminated while undergoing the current logic's stipulations.

One of the challenges I faced within this research was understanding the code. There were about 50 different files that had little to no documentation other than a thesis and comments within the code, which is common with most code in the digital world. In addition, there were poor variable names such as "ac" and "alfd" that I had to go back through the code to understand what each variable meant until it was ingrained in memory. By the power of perseverance and

patience, I was able to further document important functions such as the decision logic step-by-step to allow for others to further build upon these content-based oracles.

7.0 ACKNOWLEDGEMENTS

This research was supported by an Honors College Research Grant on behalf of the University of Arkansas Honors College.

I would like to thank Dr. Reid Phillips for assisting me through the comprehension of his existing content-based oracles. His knowledge contributed greatly to the progression of this research.

I would like to thank Dr. Wing-Ning Li for being my mentor throughout this research. His guidance and patience were crucial to this research.

8.0 REFERENCES

[1] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh and A. Byers, "Big data: The next frontier for innovation, competition, and productivity.", 2011.

[2] S. Ansari, R. Mohanlal, J. Poncela, A. Ansari and K. Mohanlal, ""Importance of Big Data"," 2015.

[3] "Big Data Analytics: What it is and why it matters," [Online]. Available: https://www.sas.com/en_us/insights/analytics/big-data-analytics.html.

[4] R. Phillips, P. Benham, W. N. Li, G. Beavers and C. Thompson, "“Automating File Schema Recognition Via Content-Based Oracles”," in *Proc. 2008 International Conference on information and knowledge engineering*, 2008.

[5] R. A. Phillips, W. N. Li, C. Thompson and W. Deneke, "Data File Layout Inference Using Content-Based Oracles.", Fayetteville, AR: Computational Science and Engineering (CSE), 2013 IEEE 16th International Conference on. IEEE, 2013.

[6] R. Phillips, W. N. Li and C. Thompson, "“Layout Inference: Expanded Results and Future Direction”," in *Acxiom Laboratory for Applied Research (ALAR) 2010 Conference on Applied Research in Information Technology*, 2010.

[7] R. Phillips, P. Benham, W. N. Li, G. Beavers and C. Thompson, "“A Content-Oracle Based Approach for Automating Text File Layout Inference”," Acxiom Laboratory for Applied Research (ALAR) 2008 conference on Applied Research in Information Technology, pp. 100-105.

[8] N. Blaylock and J. F. Allen, ""Statistical Goal Parameter Recognition"," *ICAPS*, vol. 4, pp. 297-304, 2004.

[9] P. Lison, "An introduction to machine learning", 2015.

[10] "Character encodings for beginners," 16 04 2015. [Online]. Available: https://www.w3.org/International/questions/qa-what-is-encoding.

[11] F. Yergeau, "UTF-8, a transformation format of ISO 10646", 2003.

[12] R. Phillips, W. N. Li, C. Thompson and W. Deneke, ""Data File Layout Inference Using Content-Based Oracles"," in *Second International Conference on Big Data Science and Engineering (BDSE)*, Sydney, Australia, December 3-5, 2013.

[13] R. Phillips, P. Benham, W. N. Li, G. Beavers and C. Thompson, "“A Statistical and Combinatorial Approach to Text File Layout Inference”," in *The Consortium for Computing Sciences in Colleges Mid-South conference*, 2008.

[14] ". f. d. d. t. fields", ACL Analytics Documentation, [Online]. Available: http://docs.acl.com/acl/11/index.jsp?topic=%2Fcom.acl.user_guide.help%2Ftable_definition%2Fc_guidelines_for_defining_delimited_text_files.html.

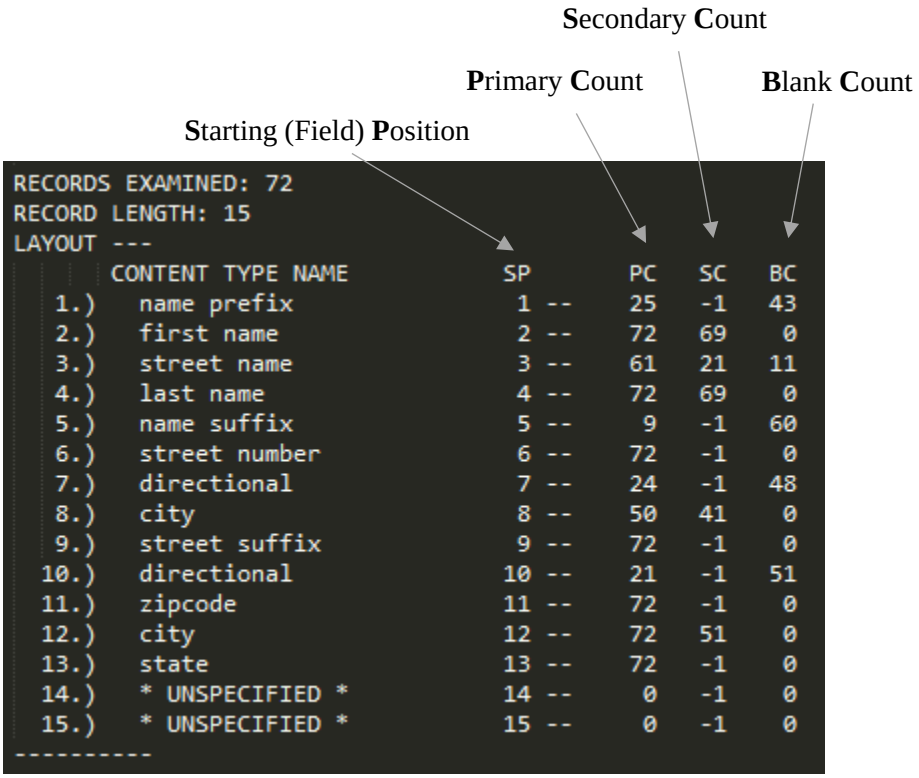
[15] R. Phillips, W. N. Li, G. Beavers, C. Thompson, J. Loghry and D. Nash, “Layout Inference: Using Content Type Domains To Infer Record Structure”, Acxiom Laboratory for Applied Research (ALAR) 2009 conference on Applied Research in Information Technology, 2009, pp. 10-16.

APPENDIX A – Oracle Information

	Oracle	Rank	Group	Maximum Blank Percentage	Minimum Threshold
0	NamePrefix	250	1	0.98	0.68
1	FirstName	1000	1	0.15	0.60
2	MiddleName	500	1	1.00	0.90
3	LastName	1000	2	0.15	0.60
4	NameSuffix	250	2	0.98	0.70
5	FullName	1000	3	0.15	0.60
6	StreetNumber	30	4	0.25	0.50
7	Directional	50	4	1.00	0.60
8	StreetName	750	4	0.50	0.65
9	StreetSuffix	750	4	0.80	0.60
10	AddressLineOne	1000	4	0.15	0.60
11	UnitDesignator	500	4	1.00	0.10
12	SecondaryRangeNumber / Unit Number	20	4	1.00	0.10
13	AddressLineTwo	800	4	1.00	0.15
14	AddressLine	1000	4	0.15	0.60
15	CityName	1000	8	0.10	0.60
16	State	800	8	0.10	0.60
17	Zipcode	500	8	0.20	0.60
18	PhoneNumber	250	16	0.98	0.50
19	Email	800	16	1.00	0.15
20	Boolean	75	16	0.10	0.95

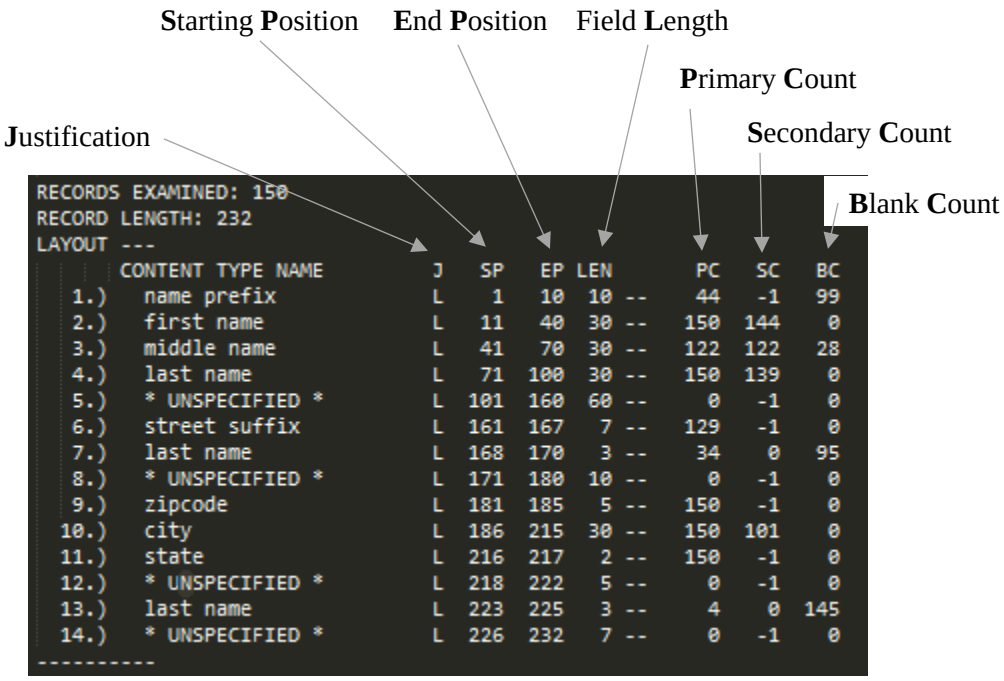
APPENDIX B – Results Output Deconstruction

Fully Delimited Files



- **Records Examined** – The number of records exampled in the sample.
- **Record Length** – The number of fields in a record.
- **Primary Count** – The number of hits within the MAXIMUM domain the corresponding oracle received for the given field position out of the number of records sampled.
    - For field 1, the name prefix oracle received 25 hits out of the 72 records sampled. These hits originated from NamePrefix_LimitedTwo.dat, the maximum domain file as specified by ConfigFile.xml.
- **Secondary Count** – The number of hits within the MINIMUM domain the corresponding oracle received for the given field position out of the number of records sampled.
    - For field 1, the name prefix oracle recorded 43 blank fields out of the 72 records sampled. This occurs when records do not contain a field. For example, not everyone has a name prefix. In these cases, name prefix would increment for every record without a name prefix.

Fully Fixed / Hybrid Files



- **Records Examined** – The number of records examined in the sample.
- **Record Length** - The length of the record is bytes.
- **Justification** – The field justification alignment.
    - For field 1, the name prefix content is justified or aligned to the left side of the file.
    - The different possibilities are L = Left, C = Center, and R = Right.
- **Starting Position** – The column number of the first symbol of the corresponding oracle.
    - For field 1, the name prefix content type begins at column 1.
    - For field 4, the last name content type begins at column 71.
- **Ending Position** – The column number of the last symbol of the corresponding oracle.
    - For field 1, the name prefix content type ends at column 10.
    - For field 4, the last name content type ends at column 100.
- **Field Length** – The length of the content type between the starting position and end position. Normally, this is index number, and not column number.
    - For field 1, the name prefix content type has a length of 10.
    - For field 4, the last name content type has a length of 30.

APPENDIX C – FullyParsedDelimited Results

Primary Count of all possibilities (not just conflicting oracles)
Columns = oracle INDEX (corresponds to Append A)
Rows = field INDEX

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	25	3		6	1				9	1		1				2					
1		72		68					72							61					
2		61		57					61							46					
3	3	62		72					70							64					
4					9		3		1	2			5				2				
5							72						72					5			
6		3		6				24								6					
7		38		56	2				68	3						50					
8		14		26						72						12					
9	12							21									6				
10							72						72					72			
11		17		31		10			59							72	1				
12					3					7		4				8	72				
13				6					3			8									
14							5	3					19								1

FullyParsedDelimited Results
FILE NAME: FullyParsedDelimited-Sample.txt
RUN TIME: 83
CHARACTER ENCODING: ASCII
FILE TYPE: 3 - fully delimited
HEADER: false
RECORD DELIMITER
  POSSIBLE: 13,10;13;10
  FOUND: 10
FIELD DELIMITER
  POSSIBLE: 9;44;124
  FOUND: 44
TEXT DELIMITER (string literals)
  POSSIBLE: 34;39
  FOUND: 39
RECORDS EXAMINED: 72
RECORD LENGTH: 15
LAYOUT ---

	Content Type Name	SP	PC	SC	BC
1	Name prefix	1	25	-1	43
2	First name	2	72	69	0
3	Street name	3	61	21	11
4	Last name	4	72	69	0
5	Name suffix	5	9	-1	60
6	Street number	6	72	-1	0
7	Directional	7	24	-1	48
8	City	8	50	41	0
9	Street suffix	9	72	-1	0
10	Directional	10	21	-1	51
11	Zipcode	11	72	-1	0
12	City	12	72	51	0
13	State	13	72	-1	0
14	* UNSPECIFIED *	14	0	-1	0
15	* UNSPECIFIED *	14	0	-1	0

POTENTIAL FIELD LOCATIONS ---

	U	CONTENT TYPE NAME	SP	PC	SC	BC
1- 1		Name prefix	1	25	-1	43
2- 1		First name	2	72	69	0
2- 2	~	First name	3	61	61	11
2- 3		First name	4	62	32	0
4- 1		Last name	2	68	32	0
4- 2	~	Last name	3	57	27	11
4- 3		Last name	4	72	69	0
4- 4		Last name	8	56	31	0

5- 1		Name suffix	5	9	-1	60
7- 1		Street number	6	72	-1	0
7- 2		Street number	11	72	-1	0
8- 1		Directional	7	24	-1	48
8- 2		Directional	10	21	-1	51
9- 1		Street name	2	72	25	0
9- 2		Street name	3	61	21	11
9- 3		Street name	4	70	59	0
9- 4		Street name	8	68	35	0
9- 5		Street name	12	59	16	0
10- 1		Street suffix	9	72	-1	0
13 -1		Unit number	5	5	-1	60
13- 2		Unit number	6	72	-1	0
13- 3		Unit number	11	72	-1	0
13- 4		Unit number	15	19	-1	53
16- 1		City	2	61	35	0
16- 2	~	City	3	46	24	11
16- 3		City	4	64	51	0
16- 4		City	8	50	41	0
16- 5		City	12	72	51	0
17- 1		State	13	72	-1	0
18- 1		Zipcode	11	72	-1	0

APPENDIX D – SmallParsed Versions

No post-directionals

1	MOTHER,Adam,Ethan,Young,PHD,567,NW,2nd,Avenue,NW,80645,La Salle,AL,Unit,R
2	MOTHER,James,Ethan,Adams,PHD,1234,NW,1st,Street,NW,80645,La Salle,AR,Unit,R
3	MOTHER,Will,Ethan,Wright,PHD,123,NW,3rd,Street,NW,80645,La Salle,CO,Unit,R
4	MOTHER,William,Ethan,Allen,PHD,654,NW,4th,Loop,NW,80645,La Salle,PA,Unit,R
5	MOTHER,Elena,Ethan,Kennedy,PHD,9876,NW,5th,Street,NW,80645,La Salle,MA,Unit,R
6	MOTHER,Isabelle,Ethan,King,PHD,98765,NW,6th,Avenue,NW,80645,La Salle,NH,Unit,R
7	MOTHER,Caroline,Ethan,Black,PHD,7412,NW,7th,Avenue,NW,NW,80645,La Salle,OH,Unit,R
8	MOTHER,Bonnie,Ethan,White,PHD,5441,NW,8th,Street,NW,NW,80645,La Salle,TX,Unit,R
9	MOTHER,Ian,Ethan,Bailey,PHD,9865,NW,9th,Circle,NW,NW,80645,La Salle,LA,Unit,R
10	MOTHER,Damon,Ethan,Brown,PHD,98765,NW,10th,Avenue,NW,80645,La Salle,TN,Unit,R
11	MOTHER,Paul,Ethan,Mathews,PHD,840,NW,11th,Street,NW,80645,La Salle,AK,Unit,R
12	MOTHER,Stephen,Ethan,Mathew,PHD,987,NW,12th,Street,NW,80645,La Salle,AR,Unit,R
13	MOTHER,Steven,Ethan,Nelson,PHD,65,NW,13th,Road,NW,80645,La Salle,TX,Unit,R
14	MOTHER,Harry,Ethan,Parker,PHD,8753,NW,14th,Road,NW,80645,La Salle,CO,Unit,R
15	MOTHER,Chris,Ethan,Patterson,PHD,657,SW,15th,Street,NW,80645,La Salle,VA,Unit,R
16	MOTHER,Ronald,Ethan,Bennett,PHD,65465,NW,16th,Avenue,NW,80645,La Salle,CA,Unit,R
17	MOTHER,Reginald,Ethan,Gilbert,PHD,7412,NW,17th,Street,NW,80645,La Salle,WA,Unit,R
18	MOTHER,Parker,Ethan,Somerhalder,PHD,6573,NW,18th,Road,NW,80645,La Salle,DC,Unit,R
19	MOTHER,Joseph,Ethan,Salvatore,PHD,98765,NW,19th,Road,NW,80645,La Salle,CO,Unit,R
20	MOTHER,Amanda,Ethan,Donovan,PHD,74325,NW,20th,Circle,NW,80645,La Salle,TX,Unit,R
21	MOTHER,Tiffany,Ethan,Potter,PHD,68752,NW,21st,Drive,NW,80645,La Salle,AR,Unit,R
22	MOTHER,Seth,Ethan,Clark,PHD,45,NW,22nd,Avenue,NW,80645,La Salle,KT,Unit,R
23	MOTHER,Joseph,Ethan,Cole,PHD,5325,NW,23rd,Street,NW,80645,La Salle,WV,Unit,R
24	MOTHER,Jeffrey,Ethan,Dixon,PHD,8766,NW,24th,Drive,NW,80645,La Salle,NC,Unit,R
25	MOTHER,Matthew,Ethan,Dickson,PHD,542,NW,25th,Avenue,NW,80645,La Salle,CO,Unit,R
26	MOTHER,Tyler,Ethan,Eastcott,PHD,8998,NW,26th,Road,NW,80645,La Salle,WA,Unit,R
27	MOTHER,Terrance,Ethan,McDonald,PHD,456,NW,27th,Avenue,NW,80645,La Salle,OR,Unit,R
28	MOTHER,Kylie,Ethan,Ranger,PHD,7412,NWeast,28th,Street,NW,80645,La Salle,TX,Unit,R
29	MOTHER,Kathy,Ethan,Wood,PHD,8523,NW,29th,Loop,NW,80645,La Salle,CA,Unit,R
30	MOTHER,Christin,Ethan,Henderson,PHD,9632,NW,30th,Avenue,NW,80645,La Salle,CO,Unit,R
31	MOTHER,Christine,Ethan,Hill,PHD,8987,NW,31th,Avenue,NW,80645,La Salle,TX,Unit,R
32	MOTHER,Patricia,Ethan,Jones,PHD,6543,NW,32th,Drive,NW,80645,La Salle,OK,Unit,R
33	MOTHER,Chris,Ethan,Hicks,PHD,3210,NW,33th,Avenue,NW,80645,La Salle,MO,Unit,R
34	MOTHER,Will,Ethan,Garcia,PHD,7410,NW,34th,Street,NW,80645,La Salle,KS,Unit,R
35	MOTHER,James,Ethan,Rodriguez,PHD,9870,NW,35th,Street,NW,80645,La Salle,FL,Unit,R
36	MOTHER,Tyler,Ethan,Rogers,PHD,98765,NW,36th,Street,NW,80645,La Salle,NV,Unit,R
37	MOTHER,Hannah,Ethan,Taylor,PHD,986,NW,37th,Road,NW,80645,La Salle,AZ,Unit,R
38	MOTHER,Chloe,Ethan,Sullivan,PHD,7589,NW,38th,Street,NW,80645,La Salle,PA,Unit,R
39	MOTHER,Jennifer,Ethan,Ward,PHD,4561,NW,39th,Avenue,NW,80645,La Salle,NH,Unit,R
40	MOTHER,Mark,Ethan,Weaver,PHD,9876,NW,40th,Circle,NW,80645,La Salle,MA,Unit,R
41	MOTHER,Bill,Ethan,Adams,PHD,74132,NW,41st,Drive,NW,80645,La Salle,ME,Unit,R
42	MOTHER,Iris,Ethan,Brown,PHD,7456,NW,42nd,Avenue,NW,80645,La Salle,CO,Unit,R

Varied pre-directionals

1	MOTHER,Adam,Ethan,Young,PHD,567,NW,2nd,Avenue,80645,La Salle,AL,Unit,R
2	MOTHER,James,Ethan,Adams,PHD,1234,N,1st,Street,80645,La Salle,AR,Unit,R
3	MOTHER,Will,Ethan,Wright,PHD,123,S,3rd,Street,80645,La Salle,CO,Unit,R
4	MOTHER,William,Ethan,Allen,PHD,654,E,4th,Loop,80645,La Salle,PA,Unit,R
5	MOTHER,Elena,Ethan,Kennedy,PHD,9876,South,5th,Street,80645,La Salle,MA,Unit,R
6	MOTHER,Isabelle,Ethan,King,PHD,98765,Northeast,6th,Avenue,80645,La Salle,NH,Unit,R
7	MOTHER,Caroline,Ethan,Black,PHD,7412,West,7th,Avenue,NW,80645,La Salle,OH,Unit,R
8	MOTHER,Bonnie,Ethan,White,PHD,5441,N,8th,Street,NW,80645,La Salle,TX,Unit,R
9	MOTHER,Ian,Ethan,Bailey,PHD,9865,E,9th,Circle,NW,80645,La Salle,LA,Unit,R
10	MOTHER,Damon,Ethan,Brown,PHD,98765,S,10th,Avenue,80645,La Salle,TN,Unit,R
11	MOTHER,Paul,Ethan,Mathews,PHD,840,W,11th,Street,80645,La Salle,AK,Unit,R
12	MOTHER,Stephen,Ethan,Mathew,PHD,987,NE,12th,Street,80645,La Salle,AR,Unit,R
13	MOTHER,Steven,Ethan,Nelson,PHD,65,NW,13th,Road,80645,La Salle,TX,Unit,R
14	MOTHER,Harry,Ethan,Parker,PHD,8753,SE,14th,Road,80645,La Salle,CO,Unit,R
15	MOTHER,Chris,Ethan,Patterson,PHD,657,SW,15th,Street,80645,La Salle,VA,Unit,R
16	MOTHER,Ronald,Ethan,Bennett,PHD,65465,Northwest,16th,Avenue,80645,La Salle,CA,Unit,R
17	MOTHER,Reginald,Ethan,Gilbert,PHD,7412,North,17th,Street,80645,La Salle,WA,Unit,R
18	MOTHER,Parker,Ethan,Somerhalder,PHD,6573,East,18th,Road,80645,La Salle,DC,Unit,R
19	MOTHER,Joseph,Ethan,Salvatore,PHD,98765,South,19th,Road,80645,La Salle,CO,Unit,R
20	MOTHER,Amanda,Ethan,Donovan,PHD,74325,West,20th,Circle,80645,La Salle,TX,Unit,R
21	MOTHER,Tiffany,Ethan,Potter,PHD,68752,N,21st,Drive,80645,La Salle,AR,Unit,R
22	MOTHER,Seth,Ethan,Clark,PHD,45,E,22nd,Avenue,80645,La Salle,KT,Unit,R
23	MOTHER,Joseph,Ethan,Cole,PHD,5325,S,23rd,Street,80645,La Salle,WV,Unit,R
24	MOTHER,Jeffrey,Ethan,Dixon,PHD,8766,W,24th,Drive,80645,La Salle,NC,Unit,R
25	MOTHER,Matthew,Ethan,Dickson,PHD,542,N.,25th,Avenue,80645,La Salle,CO,Unit,R
26	MOTHER,Tyler,Ethan,Eastcott,PHD,8998,E.,26th,Road,80645,La Salle,WA,Unit,R
27	MOTHER,Terrance,Ethan,McDonald,PHD,456,S.,27th,Avenue,80645,La Salle,OR,Unit,R
28	MOTHER,Kylie,Ethan,Ranger,PHD,7412,Southeast,28th,Street,80645,La Salle,TX,Unit,R
29	MOTHER,Kathy,Ethan,Wood,PHD,8523,Northwest,29th,Loop,80645,La Salle,CA,Unit,R
30	MOTHER,Christin,Ethan,Henderson,PHD,9632,West,30th,Avenue,80645,La Salle,CO,Unit,R
31	MOTHER,Christine,Ethan,Hill,PHD,8987,East,31th,Avenue,80645,La Salle,TX,Unit,R
32	MOTHER,Patricia,Ethan,Jones,PHD,6543,South,32th,Drive,80645,La Salle,OK,Unit,R
33	MOTHER,Chris,Ethan,Hicks,PHD,3210,North,33th,Avenue,80645,La Salle,MO,Unit,R
34	MOTHER,Will,Ethan,Garcia,PHD,7410,SE,34th,Street,80645,La Salle,KS,Unit,R
35	MOTHER,James,Ethan,Rodriguez,PHD,9870,SW,35th,Street,80645,La Salle,FL,Unit,R
36	MOTHER,Tyler,Ethan,Rogers,PHD,98765,NE,36th,Street,80645,La Salle,NV,Unit,R
37	MOTHER,Hannah,Ethan,Taylor,PHD,986,NW,37th,Road,80645,La Salle,AZ,Unit,R
38	MOTHER,Chloe,Ethan,Sullivan,PHD,7589,W,38th,Street,80645,La Salle,PA,Unit,R
39	MOTHER,Jennifer,Ethan,Ward,PHD,4561,E,39th,Avenue,80645,La Salle,NH,Unit,R
40	MOTHER,Mark,Ethan,Weaver,PHD,9876,N,40th,Circle,80645,La Salle,MA,Unit,R
41	MOTHER,Bill,Ethan,Adams,PHD,74132,S,41st,Drive,80645,La Salle,ME,Unit,R
42	MOTHER,Iris,Ethan,Brown,PHD,7456,NW,42nd,Avenue,80645,La Salle,CO,Unit,R

APPENDIX E – SmallParsed Results

SmallParsed – Varied before City Oracle changes

FILE NAME: SmallParsed-Varied.txt
RUN TIME: 1853424
CHARACTER ENCODING: ASCII
FILE TYPE: 3 - fully delimited
HEADER: false
RECORD DELIMITER
  POSSIBLE: 13,10;13;10
  FOUND: 10
FIELD DELIMITER
  POSSIBLE: 9;44;124
  FOUND: 44
TEXT DELIMITER (string literals)
  POSSIBLE: 34;39
  FOUND:
RECORDS EXAMINED: 42
RECORD LENGTH: 14
LAYOUT ---

	Content Type Name	SP	PC	SC	BC
1	* UNSPECIFIED *	1	0	-1	0
2	Middle name	2	42	40	0
3	First name	3	42	42	0
4	Last name	4	40	37	0
5	* UNSPECIFIED *	5	0	-1	0
6	* UNSPECIFIED *	6	0	-1	0
7	* UNSPECIFIED *	7	0	-1	0
8	* UNSPECIFIED *	8	0	-1	0
9	City	9	38	3	0
10	Zipcode	10	39	-1	0
11	City	11	39	39	0
12	State	12	38	-1	0
13	* UNSPECIFIED *	13	0	-1	0
14	* UNSPECIFIED *	14	0	-1	0

POTENTIAL FIELD LOCATIONS ---

	CONTENT TYPE NAME	SP	PC	SC	BC
2- 1	Middle name	2	42	40	0
2- 2	First name	3	42	42	0
2- 3	First name	4	27	13	0
4- 1	Last name	2	35	10	0
4- 2	Last name	4	40	37	0
7- 1	Street number	6	42	-1	0
7- 2	Street number	8	42	-1	0
7- 3	Street number	10	39	-1	0

8- 1	Directional	7	42	-1	0
9- 1	Street name	2	42	6	0
9- 2	Street name	3	42	0	0
9- 3	Street name	4	38	26	0
9- 4	Street name	8	39	39	0
9- 5	Street name	11	39	39	0
10- 1	Street suffix	9	42	-1	0
13- 1	Unit number	6	42	-1	0
13- 2	Unit number	7	15	-1	0
13- 3	Unit number	8	42	-1	0
13- 4	Unit number	10	39	-1	0
13- 5	Unit number	14	39	-1	0
16- 1	City	3	42	42	0
16- 2	City	4	34	31	0
16- 3	City	9	38	3	0
16- 4	City	11	39	39	0
17- 1	State	12	38	-1	0
18- 1	Zipcode	10	39	-1	0

SmallParsed – Varied after City Oracle changes

FILE NAME: SmallParsed-Varied.txt
RUN TIME: 906
CHARACTER ENCODING: ASCII
FILE TYPE: 3 - fully delimited
HEADER: false
RECORD DELIMITER
  POSSIBLE: 13,10;13;10
  FOUND: 10
FIELD DELIMITER
  POSSIBLE: 9;44;124
  FOUND: 44
TEXT DELIMITER (string literals)
  POSSIBLE: 34;39
  FOUND:
RECORDS EXAMINED: 42
RECORD LENGTH: 14
LAYOUT ---

	Content Type Name	SP	PC	SC	BC
1	* UNSPECIFIED *	1	0	-1	0
2	Middle name	2	42	40	0
3	First name	3	42	42	0
4	Last name	4	40	37	0
5	* UNSPECIFIED *	5	0	-1	0
6	Street number	6	42	-1	0
7	Directional	7	42	-1	0
8	* UNSPECIFIED *	8	0	-1	0
9	Street suffix	9	42	-1	0
10	Zipcode	10	39	-1	0
11	City	11	39	39	0
12	State	12	38	-1	0
13	* UNSPECIFIED *	13	0	-1	0
14	* UNSPECIFIED *	14	0	-1	0

POTENTIAL FIELD LOCATIONS ---

	CONTENT TYPE NAME	SP	PC	SC	BC
2- 1	Middle name	2	42	40	0
2- 2	First name	3	42	42	0
2- 3	First name	4	27	13	0
4- 1	Last name	2	35	10	0
4- 2	Last name	4	40	37	0
7- 1	Street number	6	42	-1	0
7- 2	Street number	8	42	-1	0
7- 3	Street number	10	39	-1	0
8- 1	Directional	7	42	-1	0

9- 1	Street name	2	42	6	0
9- 2	Street name	3	42	0	0
9- 3	Street name	4	38	26	0
9- 4	Street name	8	39	39	0
9- 5	Street name	11	39	39	0
10- 1	Street suffix	9	42	-1	0
13- 1	Unit number	6	42	-1	0
13- 2	Unit number	7	15	-1	0
13- 3	Unit number	8	42	-1	0
13- 4	Unit number	10	39	-1	0
13- 5	Unit number	14	39	-1	0
16- 1	City	3	42	42	0
16- 2	City	4	34	31	0
16- 3	City	11	39	39	0
17- 1	State	12	38	-1	0
18- 1	Zipcode	10	39	-1	0
