ISBN 978-83-60810-14-9
ISSN 1896-7094

Application of Clustering and Association Methods in Data Cleaning

Lukasz Ciszak
Institute of Computer Science
Warsaw University of Technology
ul. Nowowiejska 15/19, 00-665
Warszawa, Poland
Email: L.Ciszak@ii.pw.edu.pl

***Abstract*—Data cleaning is a process of maintaining data quality in information systems. Current data cleaning solutions require reference data to identify incorrect or duplicate entries. This article proposes usage of data mining in the area of data cleaning as effective in discovering reference data and validation rules from the data itself. Two algorithms designed by the author for data attribute correction have been presented. Both algorithms utilize data mining methods. Experimental results show that both algorithms can effectively clean text attributes without external reference data.**

I. Introduction

NOWADAYS , information is the most important asset of the majority of companies. Well-maintained information systems allow the company to make successful business decisions. On the other hand, if the information is incomplete or contains errors, it may lead the company to financial loss due to incorrect strategic or tactical decisions.

The purpose of this article is to present current situation in the area of data cleaning and discuss possibilities of application data mining methods in it. The structure of this article is following: first chapter focuses on the area of data quality and data cleaning and presents a categorization of data quality issues. In its last section related work and research in this area is briefly presented. The second chapter of this article contains a discussion of possible applications of data mining methods in the area of data cleaning. The core of this chapter is the presentation of two heuristic data cleaning algorithms designed by the author that utilize a data mining approach. This chapter also contains the results of the experiments performed using the algorithms and a discussion of possible improvements.

A. Data Quality and Data Cleaning

As information is defined as data and method for its interpretation, it is only as good as the underlying data. Therefore, it is essential to maintain data quality. High quality data means that it is “fit for use”[11] and good enough to satisfy a particular business application. The following data quality measures allow one to quantify the degree to which the data is of high quality, namely:

- completeness: all the required attributes for the data record are provided,
- validity: all the record attributes have values from the predefined domain,
- consistency: the record attributes do not contradict one another; e.g. the ZIP code attribute should be within the range of ZIP codes for a given city,
- timeliness: the record should describe the most up-to-date state of the real-world object it refers to. Moreover, the information about an object should be updated as soon as the state of the real world object changes,
- accuracy: the record accurately describes the real world object it refers to; all the important features of the object should be precisely and correctly described with the attributes of the data record,
- relevancy: the database should contain only the information about the object that are necessary for the purpose they were gathered for,
- accessibility and interpretability: the metadata describing the sources of the data in the database and transformations definitions it has undergone should be available immediately when it is needed.

In most cases it is almost impossible to have only “clean” and high-quality data entered into the information system. According to the research report of The Data Warehousing Institute (TDWI), “25% of critical data within Fortune 1000 companies will continue to be inaccurate through 2007. Poor quality customer data costs U.S. business an estimated $611 billion dollars a year in postage, printing, and staff overhead” [2].

Data cleaning, also known as data cleansing and data scrubbing, is the process of maintaining the quality of data. The data cleaning solution should involve discovering erroneous data records, correcting data and duplicate matching.

Data cleaning has two main applications: Master Data Management[10][12] solutions and data warehouses [10] [11], but is also often used in transactional systems.

B. Data Quality Problems

According to [14], data quality issues may be divided into two main categories: issues regarding data coming from one source and issues regarding data from multiple sources. Both main categories may be further divided into subcategories: data quality issues on the instance and on the record level.

1) One data source – schema level issues

This type of data issues is caused in most cases by source database design flaws. If the source table does not have a primary key constraint that uses a unique record identifier, we may have two records referring to separate real world objects bearing the same identifier. If a domain check constraint is missing, a column may contain values outside a specified range, e.g. more than two values describing sex. If a not-null constraint does not exist, a record may lack a value for a mandatory attribute.

If a source schema does not have reference constraint, we may have records in one table referencing non-existent records in another table.

2) One data source – record level issues

This type of issues is related to the errors that occur at the point of data entry into the source system. Typical errors involve:

- misspellings – caused either by OCR imperfections, e.g. '1' (digit one) may be replaced with 'l' (lower case L), typos, or phonetic errors,
- default values for mandatory attributes,
- inconsistencies (e.g. incorrect ZIP code for a city),
- misfielded values, i.e. correct values but placed in wrong attributes, e.g. country="Warsaw",
- duplicate records: more than one record referring the same real world object.

3) Multiple data sources – schema level issues

Multiple source schema level errors are caused by different data models in each of the source systems. For example, there may be homonyms or synonyms at the attribute level (attributes with the same name, but different meaning or attributes with different names but the same meaning), different levels of normalization, different constraints, or different data types.

4) Multiple data sources – record level issues

This kind of data quality issues is the cross-product of all the aforementioned issues. In addition, there may be other problems:

- different units of measure for the same attribute (meters vs. inches, $ vs. €, etc.),
- Different constraints/domains; e.g. `Sex= {M, F}` ; `Sex= {0, 1}` , etc.,
- Different levels of aggregation: daily vs. weekly, monthly vs. annually,
- Duplicate records - the same object from the real world may have different representations in different source systems.

C. Data cleaning areas and related work

The following primary areas of data cleaning may be distinguished, namely:

- Duplicate matching: in case of integrating multiple sources it may happen that one or more sources contain records denoting the same real world object. The records may have various degrees of data quality. Therefore, one of the tasks of the data cleaning solution is to identify duplicates and join them into a single record whose data quality would be high. This problem is known as "merge/purge" or record linkage problem. Duplicate matching is also used to discover duplicates on the higher level, e.g. records of people that share the same address. This problem is known as household detection [14]. The research in this area (e.g., [3][14] [15][18]) is focused on devising methods that are both effective, i.e. result in high number of correct matches and low number of incorrect matches, and efficient, i.e. performing within the time constraints defined in the system requirements.
- Data standardization and correction: in case of different domains used in different source systems the data cleaning solution should transform all the values used in those system into one correct set of values used in the target system. Moreover, if any incorrect values appear, the role of the data cleaning is to identify those values and alter them. Works that concern this issue use various methods ranging from statistical data cleaning[8] to machine learning[4][12].
- Schema translation: as source systems may utilize different data models, the task of data cleaning solution is to provide a mapping from those data models to the target data model. This may require splitting free-form fields (e.g. "address line 1") into an atomic attribute set ("street"," home no", "zip code"). The research in this field focuses on devising methods that are capable of performing this process automatically [6]

II. Application of Data Mining Methods in Data Cleaning

All current data cleaning solutions are highly dependent on human input. The deliverable of the profiling phase - the first phase of a data quality assessment [13], is a set of metadata describing the source data which is then used as an input for the creation of data validation and transformation rules. However, the validation rules have to be confirmed or designed by a business user who is an expert in the business area being assessed. It is not always easy or straightforward to create such a set of business rules. The situation is very similar where duplicate matching is concerned. Even if business rules for record matching are provided, e.g. "Equal SSN's and dates of birth", it may be impossible to match duplicate records, as any of the data quality issues may occur thus preventing from exact matching. Therefore, if incorrect SSN's or dates of birth stored in different positional systems occur, exact-matching business rules may not mark the records as duplicates. As far as attribute standardization and correction is concerned, data cleaning solutions are only as good as the reference data they use. Reference data is a set of values which are considered to be valid for a given attribute, e.g. list of states, countries,car models, etc.

The aim of research led by the author of this article was to examine if data mining methods may be used to solve tasks mentioned in the above paragraph. As data mining is the process of discovering previously unknown knowledge from data, it can be used to discover data validation rules and reference data directly from the data set. Table I contains possible application of data mining methods within the area of data cleaning..

The main focus of this article is attribute correction. Other data cleaning tasks are subject to further research. This article presents two applications of data mining techniques in the area of attribute correction: context-independent attribute correction implemented using clustering techniques and context-dependent attribute correction using associations. The next sections of this chapter present two algorithms created by the author that are supposed to examine the possibility of application of data mining techniques in data cleaning. Both algorithms are intended for use in the application of text-based address and personnel data cleaning.

A. *Context independent attribute correction*

As mentioned in the previous section, attribute correction solutions require reference data in order to provide satisfying results. The algorithm described in this section is used to examine if a data set may be a source of reference data that could be used to identify incorrect entries and enable to correct them. The algorithm aims at correcting text attributes in address and personal data.

Context-independent attribute correction means that all the record attributes are examined and cleaned in isolation without regard to values of other attributes of a given record.

The key idea of the algorithm is based on an observation that in most data sets there is a certain number of values having a large number of occurrences within the data set and a very large number of attributes with a very low number of occurrences. Therefore, the most-representative values may be the source of reference data. The values with low number of occurrences are noise or misspelled instances of the reference data. Table II shows an excerpt from the distribution with values sorted alphabetically and then descending according to the number of occurrences. The attribute being examined comes from the sample data set used in this experiment and is derived from an Internet survey. As it may be noticed, there are many (6160) records that have the value "Warszawa" and very few that have a value that closely resembles "Warszawa". Therefore, the value "Warszawa" is most likely the correct value of the attribute and should enter the reference data set while the remaining values are misspelled and should be corrected to "Warszawa".

The aim of the algorithm is to create a reference data set and a set of substitution rules that could be used to correct misspelled values.

Table II .
Location attribute distribution

Location	Number of occurrences
Warszawa	6160
Warszwa	8
Warszaw	4
Warsawa	1
Warszawaa	1
Warzawa	1
Warzsawa	1

1) *Algorithm definition*

The algorithm utilizes two data mining techniques: clustering and classification using the "k-nearest neighbours" method.

The algorithm has two parameters: distance threshold - *distThresh* being the minimum distance between two values allowing them to be marked as similar, and occurrence relation – *occRel*, used to determine whether both compared values belong to the reference data set. This parameter was introduced because it is possible that there are two values that may seem similar terms of the distance but that may represent two different objects of the real world that should create separate entries in the reference data set.

To measure the distance between two values a modified Levenshtein distance [5] was used. Levenshtein distance for

Table I.
Location attribute distribution

	Clustering	Association rules	Classification
Data standardization/attribute correction	X	X	
Duplicate matching	X		
Validation rules generation			X
Outlier detection	X		
Missing attribute prediction		X	X

two strings is the number of text edit operations (insertion, deletion, exchange) needed to transform one string into another.

The algorithm for attribute correction described here utilizes a modified Levenshtein distance defined as

$$\hat{Lev}(s_{1,}s_2)=\frac{1}{2}\cdot(\frac{Lev(s_{1,}s_2)}{\|s_1\|}+\frac{Lev(s_{1,}s_2)}{\|s_2\|}) \quad (1)$$

where $Lev(s_1,s_2)$ denotes a Levenshtein distance between strings s_1 and s_2. The modified Levenshtein distance for two strings may be interpreted as an average fraction of one string that has to be modified to be transformed into the other. For instance, the Levenshtein distance between “Warszawa” and “Warzsawa” is 2, while the modified Levenshtein distance for “Warszawa” and “Warzsawa” is 0.25. The modification was introduced to be independent of the string length during the comparison.

The algorithm consists of following steps:

1. Preliminary cleaning – all attributes are transformed into upper case and all the non-alphanumeric characters are removed
2. The number of occurrences for all the values of the cleaned data set is calculated.
3. Each values is assigned to a separate cluster. The cluster element having the highest number of occurrences is denoted as the cluster representative.
4. The cluster list is sorted descending according to the number of occurrences for each cluster representative.
5. Starting with first cluster, each cluster is compared with the other clusters from the list in the order defined by the number of cluster representative occurrences. The distance between two clusters is defined as the modified Levenshtein distance between cluster representatives
6. If the distance is lower than the *distThresh* parameter and the ratio of occurrences of cluster representative is greater or equal the *occRel* parameter, the clusters are merged.
7. If all the clustered pairs are compared, the clusters are examined whether they contain values having distance between them and the cluster representative above the threshold value. If so, they are removed from the cluster and added to the cluster list as separate clusters.
8. Steps 4-7 are repeated until there are no changes in the cluster list i.e. no clusters are merged and no clusters are created.
9. The cluster representatives form the reference data set, and the clusters define transformation rules – for a given cluster cluster elements values should be replaced with the value of the cluster representative.

As far as the reference dictionary is concerned, it may happen that it will contain values where the number of occurrences is very small. These values may be marked as noise and trimmed in order to preserve the compactness of the dictionary.

2) Results

The algorithm was tested using a sample data set derived from an Internet survey. The data record is define as a tuple {*First Name*, *Last Name*, *Occupation*, *Location*}. There were about 30057 records divided into 6 batches of 5 thousand records. The attribute that was the source of the data for cleaning was “Location”. During review 1166 elements – 3.88% of the whole set, were identified as incorrect and hence subject to alteration.

Table III contains example transformation rules discovered during the execution of the algorithm

Table III.
Example transformation rules

Original value	**Correct value**
Warszwa	Warszawa
Warszaw	Warszawa
Warsawa	Warszawa
Warzsawa	Warszawa

In order to test the correctness of the algorithm, following measures are used:

- p_c – percentage of correctly altered values
- p_i – percentage of incorrectly altered values
- p_0 – percentage of values marked during the review as incorrect, but not altered during the cleaning process.

The measures are defined as

$$p_c=\frac{n_c}{n_a}\cdot 100$$
$$p_i=\frac{n_i}{n_a}\cdot 100 \quad (2)$$
$$p_0=\frac{n_{00}}{n_0}\cdot 100$$

where n_c is the number of correctly altered values, n_i the number of incorrectly altered values, n_a the total number of altered values, n_{00} the number of elements initially marked as incorrect that were not altered during the cleaning process and n_0 the number of values identified as incorrect.

Fig 1 and Table IV show the dependency between the *distThresh* threshold parameter and defined measures.

It can be observed that the percentage of values altered is growing with the increase of the *distThresh* parameter. It is caused by decreasing the restrictiveness of the algorithm in assigning values as similar. The percentage of correctly altered values reaches its highest values of 92.63% at 0.1, however, for that value of the parameter only about 7% of previously identified incorrect entries are corrected. The percentage of incorrectly altered values is also growing due to the greater tolerance for duplicate matching criteria. However, it is possible to define a value of the parameter to achieve the optimal ratio of the correctly altered values avoiding a large number of incorrect values. In the case of the data examined the optimal value of the attribute was 0.2 where 20% of incorrect entries were altered, 79.52% of which were altered correctly. What is still a subject to further experiments is the optimal value of the *occRel* parameter. Also the clustering method used here could be replaced by other clustering methods which could improve the number of correctly altered values, but might have negative influence on the cleaning execution time, as method used here does not require comparison between all the values from the cleaned data set.

The algorithms displays better performance for long strings as short strings would require higher value of the parameter to discover a correct reference value. However, as it was noted in the previous paragraphs, high values of the *distThresh* parameter results in larger number of incorrectly altered elements.

This method produces as 92% of correctly altered elements which is an acceptable value. The range of the applications of this method is limited to elements that can be standardized for which reference data may exist. Conversely, using this method for cleaning last names could end with a failure.

The major drawback of this method is that may classify as incorrect a value that is correct in context of other attributes of this record, but does not have enough occurrences within the cleaned data set.

Table IV.
Dependency between the measures and the *distThresh* parameter for context-independent algorithm

distThresh	p_c	p_i	p_0
0	0.0	0.0	100.0
0.1	92.63	7.37	92.45
0.2	79.52	20.48	36.96
0.3	67.56	32.44	29.25
0.4	47.23	52.77	26.93
0.5	29.34	70.66	23.41
0.6	17.36	82.64	19.04
0.7	7.96	92.04	8.92
0.8	4.17	95.83	1.11
0.9	1.17	98.83	0.94
1.0	0.78	99.22	0

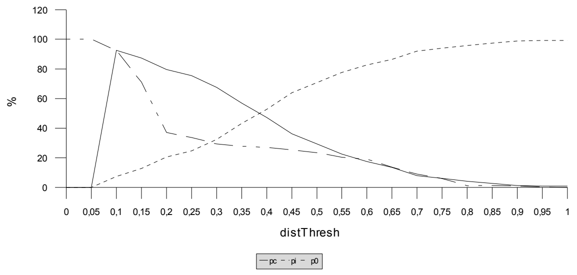

Fig 1: Dependency between the measures and the *distThresh* parameter for context-independent algorithm.

B. Context-dependent attribute correction

The context-dependent attribute correction algorithm is the second of the algorithms designed by the author that utilizes data mining methods. Context-dependent means that attribute values are corrected with regard not only to the reference data value it is most similar to, but also takes into consideration values of other attributes within a given record. The idea of the algorithm is based on assumption that within the data itself there are relationships and correlations that can be used as validation checks. The algorithm generates association rules from the dataset and uses them as a source of validity constraints and reference data.

1) Algorithm definition

The algorithm uses association rules methodology to discover validation rules for the data set. To generate frequent itemsets the Apriori[17] algorithm is utilized.

The algorithm described in this chapter has two parameters *minSup* and *distThresh*. The first of the parameters - *minSup*, is defined analogically to the parameter of the same name for the Apriori algorithm used here. The other parameter – *distThresh*, is the minimum distance between the value of the “suspicious” attribute and the proposed value being a successor of a rule it violates in order to make a correction.

Although association rules algorithms normally have two parameters: *minSup* and *minConf*, i.e. minimal confidence for generated rules, the algorithm used here does not use the latter of the two. However, the algorithm can be modified to complete the missing attribute values. In such cases the *minConf* can be used to determine the minimal confidence the rule must have in order to fill in the missing value.

For calculating distances between textual attributes the modified Levenshtein distance described in previous section is used.

The algorithm has following steps:

1. Generate all the frequent sets, 2-sets, 3-sets and 4-sets.
2. Generate all the association rules from the sets generated in the previous step. The rules generated may have 1, 2, or 3 predecessors and only one successor. The association rules generated form the set of validation rules.
3. The algorithm discovers records whose attribute values are the predecessors of the rules generated with an attribute whose value is different from the successor of a given rule. These records are marked “suspicious”.
4. The value of the attribute for a “suspicious” row is compared to all the successors of all the rules it violates. If the relative Levenshtein distance is lower than the distance threshold, the value may be corrected. If there are more values within the acceptable range of the parameter, a value most similar to the value of the record is chosen.

2) Results

The algorithm was run on a set of 287198 address records. The data records are tuples defined as `{street, location, zip code, county, state}`.

The rule-generation part of the algorithm was performed on the whole data set. The attribute correction part was performed on a random sample of the dataset consisting of 2851 records. During a review 399 attribute values were identified as incorrect for this set of records .

To verify the performance of the algorithm, measures p_c – the percentage of correctly altered values, p_i – the percentage of incorrectly altered values, and p_0 – the percentage of values not altered, as defined in previous section are used.

Table V and Fig 2 show the relationship between the measures and the *distThresh* parameter. The *minSup* parameter was arbitrarily set to 10.

Table V.
Dependency between *distThresh* parameters and measures for context dependent algorithm

distThresh	p_c	p_i	p_0
0	0.00	0.00	100.00
0.1	90	10	73.68
0.2	68.24	31.76	46.62
0.3	31.7	68.3	36.09
0.4	17.26	82.74	33.83
0.5	11.84	88.16	31.33
0.6	10.2	89.8	31.08
0.7	9.38	90.62	30.33
0.8	8.6	91.4	28.82
0.9	8.18	91.82	27.32
1.0	7.77	92.23	17.79

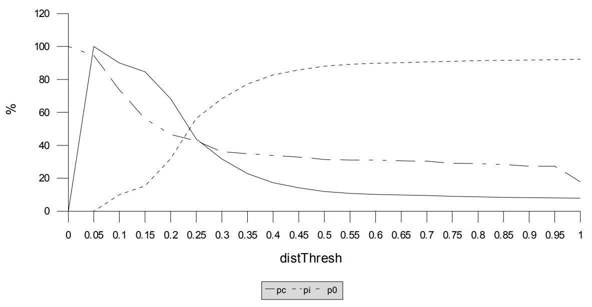

Fig 2: The dependency between the distThresh parameter and measures for context-dependent algorithm.

The results show that the number of values marked as incorrect and altered is growing with the increase of the *distThresh* parameter. Contrary to the same observation in case of context-independent cleaning, this number never reaches 100%. This proves that some attributes that may at first glance seem incorrect, are correct in the context of other attributes within the same record. The percentage of correctly marked entries reaches its peak for the *distThresh* parameter equal to 0.05. The result is better than in the case of context-independent cleaning as the number of correctly altered values for this value of the parameter is equal to the total number of altered values. This also proves that context-dependent cleaning algorithm performs better at identifying incorrect entries. The number of incorrectly altered values is growing with increase of the parameter. However, a value of the *distThresh* parameter can be identified that gives optimal results, i.e. the number of correctly altered values is high and the number of incorrectly altered values is low. In case of this experiment the value of the parameter is 0.15.

Some areas of improvement for this method may be identified. A possible change in the algorithm could involve adding one more parameter – the *minConf* for generated rules. This parameter has the same meaning as the *minConf* parameter of the Apriori algorithm. This would enable pruning the "improbable" rules and limit the number of incorrectly altered values. Also generating the rules using cleaned data would result in better algorithm performance.

III. Conclusion

The results of the experiments verifying correctness of both algorithms for attribute correction prove that using data mining methods for data cleaning is an area that needs more attention and should be a subject of further research.

Data mining methodologies applied in the area of data cleaning may be useful in situations where no reference data is provided. In such cases this data can be inferred directly from the dataset.

Experimental results of both algorithms created by the author show that attribute correction is possible without an external reference data and can give good results. However, all of the methods described here definitely necessitate more research in order to raise the ratio of correctly identified and cleaned values. As it was discovered in the experiments, the effectiveness of a method depends strongly on its parameters. The optimal parameters discovered here may give optimal results only for the data examined and it is very likely that different data sets would need different values of the parameters to achieve a high ratio of correctly cleaned data.

The above experiments utilized only one string matching distance (Levenshtein distance) was used. It is possible that other functions could result in better output and this should be explored in future experiments.

Moreover, further research on application of other data mining techniques in the area of data cleaning is planned.

References

[1] R. Agrawal, T. Imielinski, A. Swami "Mining Association Rules between sets of Items in Large Databases" in *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp.207-216

[2] B. Beal "Bad Data Haunts the Enterprise" in *Search CRM*, http://searchcrm.techtarget.com/news/article/0,289142,sid11_gci965128,00.html

[3] M. Bilenko, R. Mooney "Adaptive Duplicate Detection Using Learnable String Similarity Measures" in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2003)*

[4] T. Churches, P. Christen, K. Lim, J. Xi Zhu "Preparation of name and address data for record linkage using hidden Markov models", *BMC Medical Informatics and Decision Making*, 2, 2002

[5] W. Cohen, P. Ravikumar, S. Fienberg "A Comparison of String Distance Metrics for Name-Matching Tasks" in *Proceedings of the IJCAI-2003*

[6] W. Cohen "Integration of Heterogeneous Databases without Common Domains Using Queries Based Textual Similarity" in *Proceedings of the 1998 ACM SIGMOD international conference on Management of data* pp. 201-212

[7] M. Ester, H. Kriegel, J. Sander, X. Xu "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise" in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*

[8] B. Fung, K. Wang, M. Ester, F. Fraser "Hierarchical Document Clustering" http://www.cs.sfu.ca/~ester/papers/Encyclopedia.pdf

[9] T. Herzog, F. Scheuren, W. Winkler *Data Quality and Record Linkage Techniques* New York: Springer Science+Business Media, 234pp., ISBN: 978-0-387-69502-0

[10] R. Kimball, M. Ross *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling* Wiley, John & Sons, Incorporated, 464pp, ISBN-13: 9780471200246

[11] R. Kimball, J. Caserta *The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data* Wiley, John & Sons, Incorporated, 525pp, ISBN-13: 9780764567575

[12] M. Lee, T. Ling, W. Low "IntelliClean: A knowledge-based intelligent data cleaner" in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp.290-294

[13] A. Maydanchik *Data Quality Assessment* Technics Publications, 336pp, ISBN-13: 9780977140022

[14] E. Rahm, H.Do "Data Cleaning. Problems and Current Approaches" in *IEEE Bulletin of the Technical Committee on Data Engineering,* Vol 23 No. 4, December 2000

[15] P. Ravikumar, W. Cohen "A Hierarchical Graphical Model for Record Linkage" in *Proceedings of the 20th conference on Uncertainty in artificial intelligence*

[16] K. Ward Church "Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text" in *Proceedings of the Second Conference on Applied Natural Language Processing*

[17] J. Webb "Association Rules" in *The Handbook of Data Mining* Nong-Ye(Ed) Lawrence Erlbaum Associates, Inc., ISBN-13: 9780805855630, 724pp , pp 25-39

[18] W. Winkler "Advanced Methods For Record Linkage" in *Proceedings of the Section on Survey Research Methods* , *American Statistical Association* , pp 467-472

[19] W. Winkler "The State of Record Linkage and Current Research Problems" in *Proceedings of the Survey Methods Section*, *Statistical Society of Canada*, pp 73-80