Unsupervised Duplicate Detection Using Sample Non-Duplicates

Vom Fachbereich Informatik
der Technischen Universität Darmstadt
zur Erlangung des akademischen Grades eines
Doktor-Ingenieurs (Dr.-Ing.)
genehmigte

Dissertation

von
Diplom-Ingenieur

Patrick Lehti

aus Darmstadt

Referent: Prof. Dr.-Ing. Erich Neuhold
Korreferent: Prof. Dr. Thomas Hofmann

Tag der Einreichung: 6. Februar 2006
Tag der mündlichen Prüfung: 17. Mai 2006

Darmstadt 2006
D17

Abstract

The problem of identifying objects in databases that refer to the same real world entity, is known, among others, as duplicate detection or record linkage. Objects may be duplicates, even though they are not identical due to errors and missing data.

Traditional scenarios for duplicate detection are data warehouses, which are populated from several data sources. Duplicate detection here is part of the data cleansing process to improve data quality for the data warehouse. More recently in application scenarios like web portals, that offer users unified access to several data sources, or meta search engines, that distribute a search to several other resources and finally merge the individual results, the problem of duplicate detection is also present. In such scenarios no long and expensive data cleansing process can be carried out, but good duplicate estimations must be available directly.

The most common approaches to duplicate detection use either rules or a weighted aggregation of similarity measures between the individual attributes of potential duplicates. However, choosing the appropriate rules, similarity functions, weights, and thresholds requires deep understanding of the application domain or a good representative training set for supervised learning approaches. For this reason, these approaches entail significant costs.

This thesis presents an unsupervised, domain independent approach to duplicate detection that starts with a broad alignment of potential duplicates, and analyses the distribution of observed similarity values among these potential duplicates and among representative sample non-duplicates to improve the initial alignment. To this end, a refinement of the classic Fellegi-Sunter model for record linkage is developed, which makes use of these distributions to iteratively remove clear non-duplicates from the set of potential duplicates. Alternatively also machine learning methods like Support Vector Machines are used and compared with the refined Fellegi-Sunter model.

Additionally, the presented approach is not only able to align flat records, but makes also use of related objects, which may significantly increase the alignment accuracy, depending on the application.

Evaluations show that the approach supersedes other unsupervised approaches and reaches almost the same accuracy as even fully supervised, domain dependent approaches.

Deutsche Zusammenfassung

Das Problem zu erkennen, dass verschiedene Datenbankeinträge sich auf das selbe reale Objekt beziehen, ist in der Literatur als "duplicate detection" (Duplikaterkennung) oder "record linkage" bekannt. Solche Datenbankeinträge können auch dann Duplikate sein, wenn sie fehlerhafte oder unvollständige Informationen enthalten.

Klassisch tritt dieses Problem bei der Erstellung von Datawarehouses auf, wo verschiedene unabhängige Datenquellen integriert werden sollen. Datawarehouses werden oft genutzt, um wichtige Geschäftsentscheidungen zu treffen, und müssen deshalb von hoher Qualität sein, d.h. möglichst frei von Duplikaten.

Webportale sind ein weiteres Anwendungsfeld für Duplikaterkennung, denn diese ermöglichen den einheitlichen Zugriff auf unabhängige Datenquellen, und Duplikate schmälern dabei den Nutzen eines solchen Portals. Die Unabhängigkeit der Datenquellen verlangt hierbei, dass die Duplikaterkennung regelmäßig bei Aktualisierungen der Daten durchgeführt wird.

Metasuchmaschinen schließlich müssen ebenfalls eine Duplikaterkennung bei der Vereinigung ihrer Ergebnisse durchführen, um den Nutzen für den Anwender zu erhöhen. Da in diesem Fall keine Kontrolle über die Datenquellen besteht, muss die Duplikaterkennung dynamisch bei jedem Zugriff erfolgen.

Existierende Ansätze zur Duplikaterkennung benutzen üblicherweise Regeln oder eine gewichtete Aggregation von Ähnlichkeitsmaßen. Das Bestimmen dieser Regeln oder Gewichte und die Auswahl geeigneter Ähnlichkeitsmaße erfordert eine sehr gute Kenntnis der Daten oder benötigt gute repräsentative Trainingsdaten. Diese zu erhalten ist mit großem Aufwand verbunden, was sich in den hohen Kosten für typische Integrationsprojekte im Datawarehouseumfeld zeigt. Für die anderen Anwendungsfelder sind diese Ansätze gar nicht oder nur sehr bedingt geeignet.

Diese Arbeit präsentiert ein Verfahren zur automatischen Duplikaterkennung für beliebige Datenquellen. Dazu werden in einem ersten Schritt

eine Menge von Kandidatduplikaten und eine Menge von repräsentativen Nicht-Duplikaten bestimmt und deren Ähnlichkeiten berechnet. Hierfür wird angenommen (und das Verfahren auf diesen Fall eingeschränkt), dass Duplikate zwischen verschiedenen und in sich annähernd duplikatfreien Datenquellen erkannt werden sollen. Die genannten Mengen werden dann mit Hilfe einer deutlich präziseren und effizienteren Variante des sog. Sorted-Neighborhood-Verfahrens bestimmt.

Die unterschiedliche Verteilung der Ähnlichkeiten bei den Kandidatduplikaten und den Nicht-Duplikate wird dann genutzt, um iterativ die Menge der Kandidatduplikate zu verfeinern. Dazu wird einerseits eine Erweiterung eines klassischen statistischen Verfahrens (das Fellegi-Sunter-Modell), als auch ein modernes Verfahren aus dem Bereich maschinelles Lernen (Support Vector Machines) evaluiert.

Um nicht nur flache Datensätze auf Duplikate überprüfen zu können, wird zusätzlich erklärt wie das Verfahren für beliebige Graph-basierte Datenmodelle erweitert werden kann. Dazu werden zusätzliche Ähnlichkeitsmaße für Beziehungen eingeführt, die die Qualtät der Duplikaterkennung deutlich verbessern können.

Die zahlreichen Experimente zeigen, dass das vorgestellte Verfahren eine wesentlich bessere Qualität der Duplikaterkennung erreicht, als andere automatische Verfahren und sogar nah an die Qualität von Verfahren mit manuell ausgewählten Trainingsdaten heranreicht.

Acknowledgements

At first I would like to thank Prof. Dr.-Ing. Erich J. Neuhold for giving me the opportunity to do this thesis at the institute. Further I thank Prof. Dr. Thomas Hofmann for doing the second review.

A very special thank goes to my advisor Peter Fankhauser, who helped me creating many new ideas in innumerable discussions and even more valuable, his empathetic motivations to overcome the naturally occurring crisis during the writing of such a thesis.

I further thank all my colleagues for all the fruitful discussions and creating an enjoyable working atmosphere and in particular my department head Thomas Risse for giving me the freedom to finish my thesis.

Last but not least, I would like to thank all my family and friends for spending time with me, which always leaves me refreshed and newly motivated. Special thanks to Laura Wenz, who additionally reviewed my thesis and ensured that it is also understandable for non-experts.

Contents

Chapter 1

Introduction

1.1 Motivation

The problem of duplicate detection is long since known and several communities have worked on it using different terminology. Pioneering in this area has been the statistics community, which calls the problem "record linkage", the database community calls it "deduplication", "the merge/purge problem" or "eliminating fuzzy duplicates", other communities like machine learning or natural language processing investigate similar problems such as "identity uncertainty", "object identification", "object consolidation", "coreference resolution" or "entity resolution".

Duplicate detection aims to find objects from various data sources that refer to the same real world entity, where these individual objects might be erroneous and incomplete. In addition, there exists no globally unique key for these objects that would allow to directly identify them as duplicates. An example duplicate record pair can be seen in Table 1.1, which shows two records about the same restaurant taken from two different data sources.

Duplicate detection occurs in several application scenarios:

Table 1.1: Duplicate records from two *Restaurant* data sets

name	address	city	cuisine
uncle nick's	747 ninth ave.	new york city	greek
uncle nick's	747 9th ave. between 50th and 51st sts.	new york	mediterranean

Data Warehouses The classic application scenario is data warehouses. Analyses and statistics on data warehouses influence business decisions, therefore data quality is of high importance. Data warehouses are populated from several data sources, which are in general heterogeneous, i.e., they use different schemas, datatypes and formats and they may contain duplicates. To this end a data cleansing module is used to homogenize the data before finally importing them into the data warehouse.

Data warehouses are populated regularly, but not in an ad-hoc manner. And there usually exists human domain experts that are able to manually configure and tune the duplicate detection process and can also manually review unclear cases. Duplicate detection as part of the data cleansing module has the following characteristics:

- offline process - performance is less critical
- large data sources
- accuracy is highly important
- clerical review is possible for unclear cases

Web Portals There exist web portals that offer users unified access to several data sources, e.g. the upcoming german web portal for computer scientist: io-port.net. In this scenario the portal operator has access to the full data sets of the content providers, which are imported into a data base at the portal to provide unified access to this data. Duplicates in this scenario reduce the quality and usefulness of the portal for the user, but still users are also able to realize duplicates on their own.

Duplicate detection for importing the data sets into the portal database has the following characteristics:

- offline process - performance is less critical
- large data sources
- accuracy is less important
- often no clerical review is possible

Meta Search Engines Meta search engines distribute a search to several other resources and finally merge the individual results. During merging they must detect and remove duplicates to provide useful results to the user. Such meta search engines typically have no access to the full individual data sources, but can only access them via a query interface. Duplicates in this scenario also reduce the usefulness of the service, but again, the user can realize duplicates on his own to some extend.

Duplicate detection for merging the individual results has the following characteristics:

- online process - performance is important
- data sources are not directly available
- accuracy is less important
- no clerical review is possible

Data Mining An emerging scenario is data mining over the web. Here an application might want to collect information about a specific topic, e.g., a person or a company from several web sites or more generally from the web. Such a data mining application must detect, whether information from different web sites really corresponds to the same entity, which is similar to duplicate detection.

Duplicate detection for information received from different web sites has the following characteristics:

- offline process - performance is less critical
- unstructured data
- accuracy depends on the application, but is in general important
- no clerical review is possible

Focus of the Thesis This thesis focus on offline duplicate detection for large data sources with high accuracy without the need for clerical review. This matches perfectly with the web portal scenario, which is also the scenario of the project SemIPort (Semantic Methods and Tools for Information Portals) [1], which has motivated this thesis. However, the developed approach can also be used in the data warehouse scenario, where it can significantly reduce the costs of the data cleansing process. The methods can also be extended to the other scenarios, which is part of the future work.

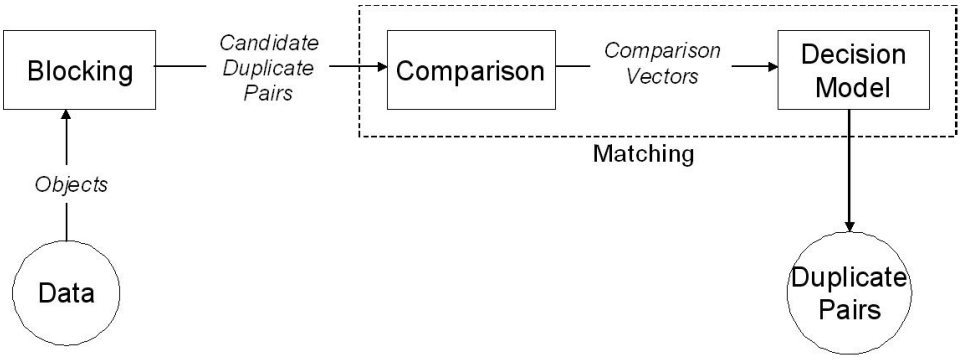

Figure 1.1: General architecture of a duplicate detection system

1.2 Problem Definition

The problem of detecting duplicates can be defined as follows: Given two sets of objects A and B divide the set of all object-pairs $(a, b) \in A \times B$ into a set of matching object-pairs M and unmatching object-pairs U. Sets are denoted by upper-case letters, individual instances by lower-case letters. An object is basically a vector of attributes (also called fields, properties or features), thus, an object-pair is a vector of attribute-pairs. Attributes of an object are denoted with a subscript specifying the label of the attribute, e.g. a_{title}.

In general the overall duplicate detection process consists of several phases as illustrated in Figure 1.1. The first phase, often called blocking, tries to efficiently find a candidate set of duplicates, then a second phase, sometimes called matching, is doing the actual duplicate decision.

The matching phase in general involves an in-depth comparison of the candidate duplicate pairs, i.e., comparing the individual attribute-pairs, which results in what is called a comparison vector $\gamma[a, b]$ for every candidate pair. The individual components of the comparison vector (γ_i) represent the comparison results between the individual attribute-pairs. These individual comparison results can be boolean (attribute matches or does not match), discrete (e.g., matches, possibly matches or does not match) or continuous (e.g., attribute values have a similarity of 0.2). The task for the matching algorithm is then to classify such a given comparison vector γ into the sets M or U by some kind of decision model. In the "Object Identification Framework" by Neiling and Jurk [60] these phases are called: "preselection", "comparison" and "classification".

The in-depth comparison of object-pairs as well as the final matching

decision are in general expensive, which explains the necessity of the blocking phase. Even moderately sized data sets with 10^5 records would result in 10^{10} comparisons for a quadratic comparison of all possible pairs. Therefore the number of such comparisons must be heavily reduced before the detailed comparison can take place.

The major problem in the blocking phase is therefore efficiency, i.e., finding a good candidate set with a minimum of resources, whereas the major problem in the matching phase is accuracy, i.e., minimizing the number of false matches and false misses.

1.3 Contribution

The contributions of this thesis are as follows:

- a method for unsupervised detection of duplicates between two data sources
- a way to automatically select a representative sample of non-duplicates that can be used by the decision models to increase their decision accuracy
- the usage of the Fellegi-Sunter model, a simple KMeans clustering algorithm and Support Vector Machines (SVMs) with such sample non-duplicates
- an extension, which allows to detect duplicates not only on flat records, but also on inter-related objects
- an extension to the sorted-neighborhood blocking method [38], which is more accurate, less costly and easier configurable than the original method and is competitive with the best other state of the art blocking methods
- extensions to the classic Fellegi-Sunter model for record linkage [33] for determining the parameters without the independence assumption and using continuous distance values; a probabilistic model that simplifies the problem of finding a good threshold on the result; the handling of null values, multi-valued attributes and relationships

1.4 Outline

The thesis is further structured as follows:

Chapter 2 describes and assesses related work in the area of duplicate detection from several communities.

Chapter 3 presents the extension to the sorted-neighborhood blocking method.

Chapter 4 presents the extensions to the Fellegi-Sunter model for record linkage.

Chapter 5 introduces the unsupervised duplicate detection approach for the matching phase using sample non-duplicates.

Chapter 6 presents the results of the experiments that show the effectiveness of the developed approach and compares it with other state of the art methods.

Chapter 7 concludes and shows potentials for future work.

Chapter 2

Related Work

Although work on the blocking problem often also addresses the matching problem and vice versa, the following sections present the proposed solutions separately, resulting in some work mentioned several times. Every subtopic is preceded by assessment criteria for this topic, which are used to make assessments about the presented approach.

2.1 Blocking

Blocking tries to efficiently find a set of candidate duplicates. A good candidate set of duplicates is a small set of pairs that contains all true duplicates. The ideal blocking algorithm finds this with a minimum of resources, i.e., it needs only very few and cheap comparisons of record-pairs, and it requires only low or no user-interaction for the configuration. In summary, a blocking method should ideally fulfill the following requirements:

- no false misses (very high recall)
- few false matches (high precision)
- few and cheap comparisons (low cost)
- little or no user-interaction (easy configuration)

It can be easily seen that there is a trade-off between "high recall", "high precision" and "easy configuration". As the precision is increased in the matching phase, but the recall can not be increased later on, the priority should definitely be on the recall.

2.1.1 Key-based Blocking

Already the pioneering work in duplicate detection of Newcombe et al. [61] in 1959 suggested the use of a blocking phase. Their approach can be categorized as a simple key based method, i.e. out of all or some of the attribute values of every record a key value is calculated by a predefined expression. All record pairs that share the same key value are then defined as candidate duplicates. Newcombe has addressed the particular problem of matching birth records to marriage records, both record files contain the husband's surname and the wife's maiden name. For the key value, a phonetic code of these names was calculated consisting of the first letter of the name followed by three numeric digits known as the Russell Soundex Code. This code is designed to remain unchanged with many of the common spelling variations and thus allows to bring together linkable records, which would have been widely separated, if arranged in a strictly alphabetic sequence.

Jaro [41] also suggests the use of key expressions for blocking. In his "AutoMatch" system [42], he extends this standard key approach by using several blocking passes with multiple keys, which shows to increase the accuracy of the resulting candidate set, i.e., it reduces the number of false matches and false misses.

Fellegi and Sunter accept the necessity of a blocking phase in their important theory for record linkage in 1969 [33] (which is presented in detail in Section 2.3.3). They formally define the impact on their duplicate detection approach when using a blocking method in terms of introduced false misses. Kelley [46] extends this work in 1985 by introducing measures of cost and benefit of a blocking scheme, where cost is again defined in terms of introduced false misses and benefit is defined in terms of the size of the resulting candidate set. For key based blocking and known match probabilities of such a key for true duplicates and non-duplicates respectively, this method can be used to select the best blocking key balancing the cost and benefit of it.

The main advantage of the key-based blocking algorithm is its low complexity and therefore its low cost. The calculation of the key values is of linear complexity and the following key matching is a simple sorting task. However, this approach has several significant drawbacks. Whenever the key value of two duplicate records is slightly different (e.g. caused by small typographical errors) the two records are put into different blocks, which can only partly be improved by using phonetic encodings. To this end, in order to achieve a high recall, it is often necessary to choose a very generic key, which results in large blocks and as a consequence in low precision.

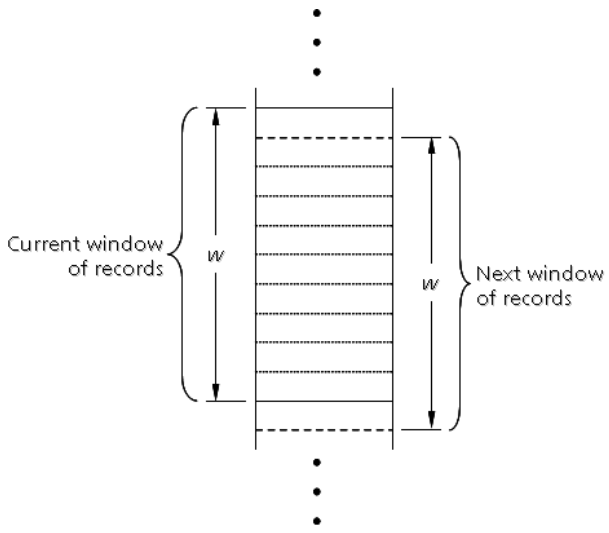

Figure 2.1: Concept of the Sorted-Neighborhood Method

A perfect key is therefore highly reliable (containing only few errors) and highly unique (only few records share the same key), which is often either not available or requires good knowledge of the domain and on the degree of error in the concrete data. The process of finding such a key is normally done iteratively, by manually examining the output of the blocking and tuning the key until the required accuracy is reached.

2.1.2 Sorted-neighborhood blocking

Hernandez and Stolfo [38] present a complete duplicate detection framework, which has a blocking algorithm as integral part. This algorithm is called the sorted-neighborhood method, it sorts the records based on a sorting key and then moves a window of fixed size sequentially over the sorted records. All records within such a window are then paired with each other and classified as candidate duplicates. Figure 2.1 shows the concept of the sorted-neighborhood method.

In contrast to the standard key-based blocking, the sorted-neighborhood method is also able to include pairs with similar keys, i.e. pairs with keys that are ordered nearby. On the other hand, if the number of records sharing the same key is larger than the fixed window size, some candidate pairs will not be classified. Large window sizes have the disadvantage that they do not only generate many candidate pairs, they also do not take into account

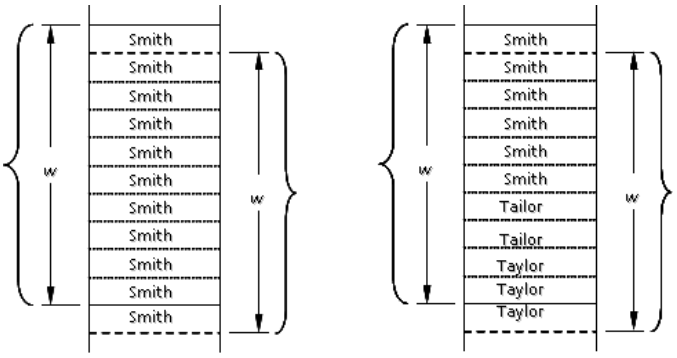

Figure 2.2: Problems of the Sorted-Neighborhood Method: a) window too small b) window too large

the similarity between the keys within the window. Therefore records with completely different keys might get into the set of candidate duplicates. Figure 2.2 shows the problems of the original sorted-neighborhood method.

Although the sorted-neighborhood method has the potential to be more robust against small typos in the key values, it also fails if typos occur in the first characters of the sorting key, which can result in the records being ordered outside the window size. For this particular problem, the authors suggest to do multiple passes with different sorting keys but small window sizes and they have shown that these multiple passes generate higher accuracy than one pass with a large window size.

The complexity of this algorithm depends on the chosen window size and is between the complexity of the key-based method, also requiring a key calculation and sorting of the records, and the fully quadratic complexity for a window size of n. Recall and precision values also depend on the window size, the larger the window the more candidate pairs are generated, potentially including more true duplicates (higher recall), but also more false duplicates (lower precision). The algorithm requires to carefully select good sorting keys and an ideal window size, which are knowledge-intensive tasks requiring high user-interaction. However, using multiple passes with different keys reduces the need for selecting a perfect key, i.e., simple keys can be selected that should only be unique enough such that there exist no record sets with the same key value, that are larger than the window size.

The blocking approach presented in this thesis is a modified version of the original sorted-neighborhood method that resolves its main drawback by replacing the fixed sized window by a dynamically sized window based

on a distance measure between the keys. The evaluation shows not only a significant increased accuracy of this variant, but also reduced overall costs of the overall duplicate detection process.

Sung et al. [74] present another variant to the sorted-neighborhood method. The main idea is to use a distance measure for the sorting key, for which the triangle inequality holds, they introduce a distance measure called "TI-Similarity", which is based on the number of characters in common. This triangle inequality allows to determine upper and lower bounds for the distance between the records A and C ($Sim(A, C)$) given the distance $Sim(A, B)$ and $Sim(B, C)$. Then during the blocking process, the current record needs only be compared with one record in the window (called the anchor record) and the distance to all other records in the window is approximated via the upper and lower bound distances. This reduces the number of expensive distance calculations. This approach is orthogonal to the approach presented in this thesis and using them together might further increase their performance.

Lee et al. [49] suggest the use of preprocessing steps, that scrub dirty data fields and then separate the data into tokens and sort the tokens alphabetically. These preprocessing steps increase the probability that similar data fields and their records are brought closer after sorting and will therefore get into the same window.

2.1.3 Advanced blocking methods

More recent work on blocking has suggested more complex methods, that are more robust to errors in the blocking key.

The "Bigram Indexing" method as implemented in the "Febrl" system [17] converts the blocking key values into a list of bigrams (sub-strings containing two characters) and builds sublists of all possible permutations of these bigrams. The resulting bigram lists are sorted and inserted into an inverted index, which is used to retrieve the corresponding record numbers in a block. Every record is inserted into several blocks, the number of such blocks depends on the length of the blocking key value and the length of the sublists. The shorter the sublists, the more sublists there will be per blocking key value, resulting in more blocks in the inverted index. In the information retrieval field, bigram indexing has been found to be robust to small typographical errors in documents.

McCallum et al. introduced the "Canopy Clustering" method [55], which forms blocks of records based on those records placed in the same canopy. Canopies are created by randomly choosing a record r, and putting r and all

other records, which are in a distance of a selected loose threshold $T1$ of the blocking key value of r into a new canopy. Then r and all records within a distance of a selected strict threshold $T2$ (where $T1 > T2$) are removed from the list of records. This is repeated until no more records are in the list. This method in general results in overlapping canopies. The used distance measure can be e.g. a TFIDF distance metric based on tokens, for which an inverted-index based retrieval system can find nearby pairs quite quickly.

Baxter et al. [5] compared the standard key-based blocking, the sorted-neighborhood, the bigram indexing and the canopy clustering with each other in terms of accuracy. Their results clearly showed that the bigram indexing and the canopy clustering significantly outperform the two more traditional approaches.

However, the main disadvantages of these two approaches are the required additional indices. The bigram indexing needs the inverted index for the bigram sub-lists, whereas the canopy clustering needs a special index for finding all records that are within the threshold of a distance measure, which depends on the used distance measure.

The approach presented in this thesis is compared to these methods by using the same experimental settings as in [5]. It shows that the approach can easily compete with even the best other blocking methods, but without the need of additional special indices.

Hylton [40] used a very simple method to find potential duplicates in the publication domain. For every record he generates three queries containing one name of an author and two words of the title and executes these queries against a full text index on all records. All found records are used as potential duplicates. Obviously such a method does not result in a high precision as these queries are very unspecific and the method is also highly domain dependent and requires manual definition of reasonable queries.

Monge and Elkan [58] present a full duplicate detection framework, but their main contribution is an efficient domain-independent algorithm, which makes their work more blocking related. Their algorithm is called priority queue. This queue contains a fixed number of record sets, which contain similar records that can be paired as candidate duplicates. The sorted list of all records is sequentially scanned and every record R_j is compared with the members R_i in the record set with the highest priority in the queue. This comparison is done with an edit-distance algorithm, called a variant of Smith-Waterman [59]. If the comparison yields a distance below some threshold T_1, the record R_j is included into the set of R_i, if the distance is higher than some threshold T_2 this record set is skipped and the comparison is continued with the next highest priority set in the queue. If no set is

found at all, a new singleton set is put into the queue. In order to reduce the number of comparisons, the record sets are pruned, i.e. they only contain "representative" members of the set, in particular non-representative members are those, which are very similar to other members. However, the authors do not make it clear how to find the two thresholds and how to decide if a record is "representative".

Recently, Jin et al. [43] used vector space submapping for duplicate detection. The basic idea is that the values of the blocking key are mapped to a multidimensional Euclidean space that preserves domain-specific similarity. There exist several mapping algorithms that can be used for this, the authors used a variant of FastMap [32]. In a second step a multidimensional similarity join over the chosen attributes is used to determine similar pairs of records. Their experiments show, that this algorithm outperforms the original sorted-neighborhood algorithm significantly. Unfortunately, no experiments on commonly used benchmarks where conducted, therefore a comparison to other blocking methods is hardly possible.

Finally, Gu and Baxter [37] find that in some domains every blocking method will generate too many candidate duplicates, i.e. produce a low precision. This happens, when some values of the blocking key are very common, e.g. when blocking on surname for an Anglo-Celtic population, "Smith" and "Taylor" are populous and result in many candidate duplicate pairs. The authors suggest to further filter such large blocks with a different filter variable, in order to reduce the number, i.e. increase the precision of the resulting candidate pairs. This approach is again orthogonal and for certain domains all blocking methods could profit of such a post-processing step.

Winkler also summarizes the state-of-the-art for blocking in [83].

2.2 Comparison Methods

A comparison method compares the individual attribute-pairs and generates a comparison vector. Therefore the major problem of a comparison algorithm is to estimate the degree of similarity between attribute values. Ideally an equivalent attribute value would always get a higher similarity score than a non-equivalent value. Therefore the requirements for a good comparison method are as follows:

- clear separation (accurate similarity estimation)
- little or no user-interaction (easy configuration)

Such a perfect similarity function is in practice hardly possible, as the cause for differences between actually equivalent values can be manifold:

Typographical Errors The easiest case are typographical errors, where the same string contains wrong, missing, additional or interchanged characters or tokens. Similarity measures for this kind of errors are numerous and well studied (see Section 2.2.1). They are in general called distance functions, measuring dissimilarity rather than similarity.

Datatype Dependency If a value is not a simple string, but some kind of primitive datatype, simple string-based similarity measures often work poorly. E.g. a year value of "2000" and "1999" do not have a single character in common and would therefore be judged as very different by a string-based similarity measure, in this case a simple arithmetic difference based similarity measure would be much more effective. However, for the same datatype year the values "2001" and "2010" would be better judged by the string-based similarity, as a simple typo is possible.

Domain Dependency The most difficult case is, where an equivalence between values can only be recognized if additional domain knowledge is available. E.g. in the publication domain the two values for the conference event "VLDB-95" and "Int. Conference on Very Large Data Bases, 1995" look very different and can only be recognized as equivalent, when knowing that "VLDB" is a common abbreviation for "Int. Conference on Very Large Data Bases".

2.2.1 String Similarity Measures

String similarity metrics measure the dissimilarity between strings that is mainly caused by typos. They can be classified into character-based functions and token-based functions.

Character-based functions are often based on the cost of edit functions that need to be performed in order to convert a string s into a string t. These distance functions are in general dissimilarity measure - the complement of a similarity measure. The most simple edit-distance function is the Levenshtein distance [50], which counts the number of character insertions, deletions or switching. More complex edit-functions are affine functions, which assign a relatively lower cost to a sequence of insertions or deletions, like a missing word or missing suffix. The distance function from Monge and Elkan [59] is an example for such an affine edit-distance functions. Other

functions, which are not directly based on an edit-distance model, but on the number and order of the common characters between two strings, are the Jaro metric [41] and its extension by Winkler called the Jaro-Winkler metric [81].

On the other hand, token-based functions regard strings as a bag of tokens. E.g. the "Jaccard" similarity simply calculates the ratio between the number of tokens in common to the number of all different tokens of the two input strings. In the information retrieval community measures known as TFIDF or cosine similarity are widely used, which considers the frequency of individual tokens within the whole corpus. That allows to assign matching tokens a lower score, which are very common like "Corp." in company names, than to matching tokens, which are more unique like "Lucent".

Cohen et al. [23] compared a large number of string distance measures and showed that the effectiveness of edit-distance vs. token-based distance depends on the data. Hybrid distance functions can be able to overcome this limitation.

Bilenko and Mooney [10, 11, 12] present a supervised learning approach, where they train a character-based string distance measure with affine gaps based on the work of Ristad and Yianilos [66] on a set of labeled duplicates and non-duplicates. They show that such a learned variant provides higher accuracy of the similarity score on the attribute-level.

Tejada et al. [75, 76] also present a supervised learning approach on a token-based distance function, which tries to apply a number of predefined token transformations, like equality, stemming, soundex, abbreviation, prefix, suffix, substring and so on. Every transformation gets a different weight or similarity score, which sums up to the overall similarity score of an attribute. The concrete weight for every transformation is actively learned from a set of labeled duplicates and non-duplicates.

2.2.2 Similarity Measures using Co-occurrences

Recently, several approaches propose new similarity measure using co-occurrence information of different values. These try to solve the problem of detecting similarities between synonyms.

Ananthakrishna et al. [2] realized that string distance metrics are not always able to detect similarities. Their example comes from the address domain, where the three strings "United States", "USA" and "UK" will not be correctly judged by any string distance function. They call this kind of errors, "equivalence errors", and suggest the use of a co-occurrence

similarity function. This function measures the co-occurrence of different values between identical tuples. E.g. if "United States" and "USA" often co-occur with the same state information ("Missouri", "Washington"), but "UK" occurs only with a distinct set of states, their (non-)equivalence can be detected. Their co-occurrence measure is basically the ratio of the number of co-occurring values to the number of overall occurring values for one of the objects.

Bhattacharya and Getoor [8, 7] present a similar distance metric as [2], but they use an iterative approach for further tuning the distance metric, when new duplicates are detected. This similarity measures counts how often two objects co-occur together with already known duplicates. They use a fixed threshold during the iteration to decide for duplicates, which may lead to many false matches, if once actual non-duplicates are wrongly declared as duplicates.

Kang et al. [45] present an approach for value mappings that allows to detect synonyms, which can not be identified using string similarity measures. They build a statistical model that covers the co-occurrences of values and find a bijective mappings from values in one source to those in another. These mappings are not able to handle generalization relationships, where one value maps to several other values in the other source and furthermore no confidence is given for the results.

Similarly, Noren et al [62] present another interesting approach using a statistical model based on co-occurrences, which is called the hit-miss model. They extend this model to also take other similarities, like string similarities, into account.

This thesis does not attempt to contribute new similarity measures, but existing similarity measures are applied and experiments show the enormous influence of such similarity measures on the final duplicate decision.

2.3 Decision Models

Similarity measures generate a comparison vector consisting of the similarities between the individual attributes. A decision model decides whether the comparison vector belongs to an actual duplicate, a non-duplicate or a possible duplicate that needs clerical review. The ideal decision model never declares a true non-duplicate as duplicate, a true duplicate as non-duplicate and the number of possible duplicates is small. Additionally, the ideal algorithm should also need only little or no manual configuration. In summary, a decision model should ideally fulfill the following requirements:

- high accuracy (high recall + high precision)
- few possible matches (few clerical reviews)
- little or no user-interaction (easy configuration)

It can be easily seen that there is again a trade-off between high recall, high precision and few clerical reviews. The higher the recall and precision for the automatically decided duplicate pairs, the higher will also be the number of unclear/possible duplicate pairs and vice versa.

The major problem for a decision algorithm is to understand the relevance of the different attributes for the duplicate decision, e.g. how to decide if one attribute matches, but another does not. Often the individual attributes do not have the same relevance, e.g. in the person domain a matching "person name" is a much stronger indication for a duplicate than a matching "year of birth", because of the higher *uniqueness* of "person name" in contrast to "year of birth". However, a non-matching "year of birth" can be a strong indication for a non-duplicate, when we assume a high *reliability* for "year of birth".

Uniqueness and reliability are the two characteristics of an attribute that make up its relevance for duplicate detection. The uniqueness is a property that is reflected in the distribution of the similarity of the attribute-pairs of non-duplicates, i.e. if an attribute is highly unique, the mean similarity on non-duplicates for this attribute will be low. The reliability is a property that is reflected in the distribution of the similarity of the attribute-pairs of duplicates, i.e. if an attribute is highly reliable, the mean similarity on duplicates for this attribute will be high. The ideal attribute is of course highly unique and highly reliable, but such attributes often do not exist in real world examples.

Further problems for a decision model arise, if the individual values depend on each other, which is typically the case.

Verykios et al. [77] mention that in some cases accuracy is not the main issue, but the resulting cost of erroneous duplicate detection, e.g., sending information to the same customer twice is not as costly as not sending information to an interested customer.

2.3.1 Trivial Decision Models

Monge and Elkan [58] ignore the relevance of the individual attributes by simply concatenating all values to a single string. The string similarity of this string is then taken as overall similarity of the whole record. This does

```
if(similar.ssns && similar.names){
  merge_tuples(person1, person2);
  continue;
}
if((similar.ssns || similar.names) &&
  very_similar_addrs){
  merge_tuples(person1, person2);
  continue;
}
...
```

Figure 2.3: Sample matching rules from [38]

not only ignore relevance, it will also fail, if some attributes may contain null values, as this will result in a poor string similarity as well. This method is used as baseline in the experiments of this thesis.

Similarly, Ananthakrishna et al. [2] concatenate the values of the direct attributes to a single string, ignoring their relevance. However, they combine the string similarity between these concatenated strings with their additional co-occurrence similarity between the related objects. The overall similarity is therefore a weighted mean between two similarity measures. Still, the drawbacks of simply concatenating the individual values remain the same.

Lee et al. [49] assign field weightings that indicate the relative importance of a field to compute the overall record similarity. The individual weights must be manually assigned to every field. Furthermore, a single weight can not capture the *uniqueness* and *reliability* feature of a field.

2.3.2 Rule-based Decision Models

Rule-based matching is a totally manual approach, where a domain expert defines a set of logic rules, that describe in which conditions a pair should be considered as duplicate. Such approaches come from the database community and are very flexible but knowledge-intensive.

The work of Hernandez and Stolfo [38], which also presents the sorted-neighborhood blocking algorithm, use a set of rules for the matching task, which are directly coded as a C program consisting of a list of conditions. Their application domain is address data, there such rules look similar like the ones shown in Figure 2.3.

That reads: if two persons have a similar ssn and similar names they

are declared as duplicate, otherwise if they have either a similar ssn or similar names and a very similar address they are declared as duplicate. Being "similar" or "very similar" is defined by a threshold on the edit-distance between the corresponding attribute values. They need 26 rules for this domain, which are developed in an iterative modify-review-process. As these rules are coded directly in C, such rules are very flexible and any possible condition and any possible similarity measure can be applied to gain a perfect accuracy. However, the definition of such rules requires perfect knowledge about the domain and extensive testing and evaluation of the rules.

Galhardas et al. [34, 35] present a framework for data cleansing based on a set of new operators for SQL. This framework contains services for normalization and mapping of tables (schema integration), duplicate detection and elimination on single tables as well as duplicate detection and merging between different tables. The matching operators are conditional rules, that define when two tuples should be considered as duplicates. These rules are very similar to those of [38], but are expressed in SQL-like expressions. It is also possible to explicitly declare, in which conditions no automatic matching decision should be performed, but a human must review a tuple.

One speciality of this approach is that they do not use a blocking phase, but their duplicate detection is directly based on the Cartesian product, that means every tuple is compared to all other tuples. This significant performance drawback is addressed by several optimization techniques, which are partly available from a query optimizer of a standard RDBMS system and are extended by additional external optimizations. This framework is very flexible, but requires intensive human interaction and knowledge in order to provide good rules.

Lee et al. [48] also suggest the usage of rules in the form of if-then statements. Additionally they compute the transitive closure over the identified duplicates, that is, if A is equivalent to B, and B is equivalent to C, then A is also equivalent to C under the assumption of transitivity. In order to not increase the false-positive error rate, which might happen, when incorrect pairs are merged by the transitive closure step, they introduce a certainty factor to every duplicate detection rule, which represents the confidence in the rule's effectiveness in identifying true duplicates.

Cohen [20] introduces the WHIRL query language to define matching rules. This query language extends SQL with a special join operator that uses a similarity comparison function based on the vector-space model commonly adopted in statistical information retrieval.

Similarly, Gravano et al. [36] present a method for approximate string

joins in a database based on q-grams, which can be used to find similar attribute values and similar records.

2.3.3 The Fellegi-Sunter Model for Record Linkage

Based on the initial ideas and problem description of Newcombe [61], Fellegi-Sunter [33] defined a theory for record linkage that takes the relevance for individual attributes into account. To this end, they define the following probabilities on the comparison vector $\gamma[a, b]$ for an object pair (a, b):

$$m(\gamma[a, b]) = P(\gamma[a, b] \mid (a, b) \in M) \tag{2.1}$$

$$u(\gamma[a, b]) = P(\gamma[a, b] \mid (a, b) \in U) \tag{2.2}$$

Here $m(\gamma[a, b])$ is the conditional probability of $\gamma[a, b]$, given that a and b are elements of M (the set of matching pairs) and $u(\gamma[a, b])$ is the conditional probability of $\gamma[a, b]$, given that a and b are elements of U (the set of unmatching pairs). They have shown that the ratio $w(\gamma[a, b]) = m(\gamma[a, b])/u(\gamma[a, b])$ can then be used as a decision function for a duplicate. They further define how to set the appropriate thresholds on $w(\gamma[a, b])$ to decide for a duplicate, non-duplicate or potential duplicate given acceptable error rates for false misses and false matches using the independence assumption. For a more general case Belin and Rubin [6] developed a method for setting the threshold.

In order to determine the parameters $u(\gamma[a, b])$ and $m(\gamma[a, b])$ often a conditional independence between the individual components of the comparison vector is assumed. Under this assumption the parameters $u(\gamma[a, b])$ and $m(\gamma[a, b])$ can be computed using m_i and u_i for the individual probabilities for the comparison vector component $\gamma_i[a, b]$:

$$m(\gamma[a, b]) = m_1(\gamma_1[a, b]) * m_2(\gamma_2[a, b])...m_k(\gamma_k[a, b]) \tag{2.3}$$

$$u(\gamma[a, b]) = u_1(\gamma_1[a, b]) * u_2(\gamma_2[a, b])...u_k(\gamma_k[a, b]) \tag{2.4}$$

In order to calculate the probabilities m_i and u_i generally a boolean comparison vector is used, the boolean comparison values "match" or "unmatch" are determined by some threshold on the similarity measure. This threshold is calibrated such that very similar values correspond to "match", whereas less similar values correspond to "unmatch". A good calibration of this threshold is crucial for the final result, which is also shown in the following example.

Table 2.1: Example records of similar restaurants

id	name	address	city	cuisine
a	uncle nick's	747 ninth ave.	new york	greek
b	uncle nick	9th ave. 747	new york	mediterranean
c	uncle nick's	4530 north lincoln ave.	chicago	greek
d	uncle jack's	440 ninth ave.	new york	american

The probabilities m_i and u_i can then be calculated by counting the number of occurrences of the specific comparison vector value in the sets M or U divided by the size of M or U.

$$m_i(\gamma_i[a,b]) = \frac{|\{(x,y) \in M \mid \gamma_i[x,y] = \gamma_i[a,b]\}|}{|M|} \tag{2.5}$$

$$u_i(\gamma_i[a,b]) = \frac{|\{(x,y) \in U \mid \gamma_i[x,y] = \gamma_i[a,b]\}|}{|U|} \tag{2.6}$$

Hereby $m_i(\text{"}match\text{"})$ can be interpreted as the reliability of the attribute i, because a highly reliable (error-free) attribute has a high match probability for a duplicate and $u_i(\text{"}unmatch\text{"})$ can be interpreted as the uniqueness of the attribute i, because a highly unique attribute has a high unmatch probability for a non-duplicate.

Example The following example illustrates how the Fellegi-Sunter model takes relevance of individual attributes into account, based on their uniqueness and reliability. Table 2.1 shows 4 similar records of restaurants given their name, address, city and cuisine. The record-pair (a, b) represent a duplicate, whereas the record-pairs (a, c) and (a, d) represent non-duplicates. It is assumed that the attribute *name* is highly unique and reliable, the attribute *address* is unique, but not reliable, the attribute *city* is reliable, but not unique and the attribute *cuisine* is neither unique nor reliable.

For simplicity it is assumed that unique attributes are equally unique and reliable attributes are equally reliable. For this example the results of m_i and u_i are assumed to have the values as shown in Table 2.2.

For generating the boolean comparison vectors a moderate threshold is assumed, which only accepts very similar values as "match", e.g., "uncle nick's" and "uncle nick". This results in the following comparison vectors, where m stands for "match" and u stands for "unmatch": $\gamma[a,b] = (m,u,m,u)$, $\gamma[a,c] = (m,u,u,m)$, $\gamma[a,d] = (u,m,m,u)$. The influence of different thresholds (strict or loose) is shown below.

Table 2.2: Example probabilities for m_i and u_i

	match	unmatch
m_{name}, m_{city}	0.95	0.05
$m_{address}$, $m_{cuisine}$	0.6	0.4
u_{name}, $u_{address}$	0.01	0.99
u_{city}, $u_{cuisine}$	0.1	0.9

Table 2.3: Results for the restaurant example

	$m(\gamma)$	$u(\gamma)$	$w(\gamma)$
$\gamma[a,b]$	0.14	$8.9e-4$	157.3
$\gamma[a,c]$	$1.1e-2$	$8.9e-4$	13.5
$\gamma[a,d]$	$1.1e-2$	$8.9e-4$	13.5

Based on the conditional independence assumption, the probabilities $m(\gamma)$, $u(\gamma)$ and $w(\gamma)$ can now be calculated for the 3 candidate duplicates. Using the previously shown equations and the values from Table 2.2, this results in the values for $m(\gamma)$, $u(\gamma)$ and $w(\gamma)$ as shown in Table 2.3. This demonstrates that the true duplicate receives a significantly higher result value than the non-duplicates.

Dependencies The conditional independence assumption only produces good results, if this assumption really holds, which is often not the case for real data. In this example the attributes address and city are not independent, as a particular street is only present in one or a few cities. That means that the probability of a matching city is 1 if the address matches also: $m_{city}(\gamma_{city} = m \mid \gamma_{address} = m) = 1$, $u_{city}(\gamma_{city} = m \mid \gamma_{address} = m) = 1$. Taking this dependency into account increases $m(\gamma[a,d])$ to $1.2e-2$, increases $u(\gamma[a,d])$ to $8.9e-3$ and decreases $w(\gamma[a,d])$ to 1.4, which is significantly less than using the conditional independence assumption.

Boolean Values The use of boolean comparison values requires a careful calibration of the threshold. This can be easily shown, when using a strict(s) threshold that only accepts identical values as "match" or using a loose(l) threshold that accepts also less similar values as "match" with the above example. Using such a strict threshold for the pair (a,b) would result in an unmatch of "uncle nick's" and "uncle nick" and in the comparison vector $\gamma_s[a,b] = (u,u,m,u)$; using the loose threshold for the pair (a,d) would

result in a match of "uncle nick's" and "uncle jack's" and in the comparison vector $\gamma_l[a,d] = (m,m,m,u)$. The results for $w(\gamma)$ are then as follows: $w(\gamma_s[a,b]) = 7.6e{-}3/8.8e{-}2 = 0.09$ and $w(\gamma_l[a,d]) = 0.22/9e{-}6 = 24444.4$. The strict threshold would result in not detecting the true duplicate (a,b), whereas the loose threshold would result in declaring the non-duplicate (a,d) as a duplicate.

Therefore the main challenge in using the Fellegi-Sunter model is to correctly determine the $m(\gamma)$ and $u(\gamma)$ values. Fellegi and Sunter themselves have shown that it is possible to compute these probabilities directly from the data under three restrictions: γ consists only of three components, these components are conditionally independent and the components are boolean (match, non-match). More generally, Winkler [79] showed how to use the EM algorithm (Expectation-Maximization) [26] to compute the $m(\gamma)$ and $u(\gamma)$ values under the independence assumption for boolean variables.

Several approaches e.g. from Winkler [80] and Larsen and Rubin [47] tried to determine the $m(\gamma)$ and $u(\gamma)$ values in cases where the conditional independence assumption does not hold. All these approaches try to explicitly model the dependencies, which only works for boolean variables. Additionally the approach in [47] requires clerical input in order to tune its models.

All these approaches based on the Fellegi-Sunter model come mainly from the statistics community and is well summarized in [81, 82].

Recently, Ravikumar and Cohen [65] address the problem of the parameter estimation in the Fellegi-Sunter model for the most general case, using continuous valued attributes and taking dependencies into account. They use a hierarchical latent variable graphical model (HGM) to model dependencies between the continuously valued individual attributes. They show that this approach outperforms methods that discretize the continuous values into boolean values and methods that rely on the conditional independence assumption. This method is most closely related to the approach of this thesis as it also does not make any restricting assumptions. However, they need to explicitly model dependencies and their approach shows significantly worse accuracy in the experiments.

2.3.4 Decision Models using Machine Learning

The machine learning community has presented several approaches using supervised learning, which always requires the manual and knowledge-intensive task of creating a good training set.

Elfeky et al. [29] claim that probabilistic record linkage models always

have the disadvantage to handle only boolean or categorical values and require a training set, a claim, which this thesis shows to be wrong. Therefore they propose to use machine learning techniques either based on supervised training of a classifier (e.g. some kind of decision model) or using unsupervised clustering methods like k-means [52]. Beside of the problem to create a training set for the supervised approach, simple clustering methods like k-means have the problem that they are only able to identify clusters that are linear separable, which is in general not the case for real world data.

Tejada et al. [75, 76] use a supervised learning method that generates rules that describe which combination of thresholds on the individual attributes are necessary to declare a pair as duplicate. Such a rule can be something like: if the similarity between the names is higher than a threshold T_1 and the similarity between the street name is higher than a threshold T_2 than declare it as duplicate. This is implemented in the Active Atlas system. On top of this they developed the Apollo system [56, 57], which incorporates secondary sources to improve the accuracy of the duplicate detection process. Such secondary sources can e.g. provide information about office locations of companies to infer that two records with unmatching location attributes might still belong to the same company.

The ChoiceMaker system by Borthwick et al. [13] also uses rules, which they call clues and then use supervised machine learning methods to assign weights to these rules and combine the rule decision with a weighted mean.

Doan et al. [27, 70] exploit constraints on different attributes in order to check the plausibility of candidate object-pairs. Examples for such constraints include "an object with age 2 cannot match an object with a salary of 200k". In [27] they use profilers that are manually built or trained using supervision, in [70] they use a generative model and a combination of the EM and the relaxation labeling algorithm.

Bilenko and Mooney [10, 11] also use a supervised learning method to combine the individual attribute similarities. They use support vector machines (SVM) for this learning task, which basically transforms input data points into a higher dimensional space, where they are then linear separable. This approach is implemented in a duplicate detection system called MARLIN. Their evaluation results on a commonly used duplicate detection benchmark is compared to the approach presented in this thesis in the evaluation chapter. In [9] they extend their approach to an online learning scenario, where the data is streaming.

Cohen and Richman [24] use a training set to learn a hypothesis function that labels pairs as matching or unmatching. For the case that not only duplicates between two data sources should be found, but duplicates within

one data source should be clustered, a confidence measure in the labeling for a match is used as edge weight in a graph that connects every potential duplicate object. A greedy agglomerative clustering algorithm is then used to find the actual clusters.

The drawback of manually selecting a good training set for supervised learning approaches is addressed by Sarawagi and Bhamidipaty [68], who present an active learning approach, which selects relevant pairs for manual labeling. They show that this leads to a significantly reduced number of required labeled pairs.

Recently, Chauduri et al. [16] present an unsupervised approach using a modified clustering algorithm introducing two additional properties for duplicates (compact set and sparse neighborhood). They evaluate this approach also on data sets that are used to evaluate the approach in this thesis: their results show significantly worse accuracies.

2.3.5 Graph-based Decision Models

Graph based decision models also build on the idea that taking relationships between different objects into account increases the overall duplicate detection accuracy. In contrast to the relationship based similarity measures, these decision models build up graph structures between the objects and then either try to propagate information about inferred duplicates through the graph or find some global optimal solution.

Pasula et al. [63] examine the problem of citation matching, which is identifying if a given citation refers to a specific paper or not. For this task they learn a relational probability model - a type of Bayesian network - and parsing rules from a given set of parsed and classified citations. On this model Markov Chain Monte Carlo is used as approximation method for actually inferencing the probability of a given citation to match a specific paper. Besides the problem of providing a good training set, it has the disadvantage of using a generative model, which tends to become extremely complex when trying to model all possible dependencies and then good parameter estimation is nearly impossible.

McCallum and Wellner improve on the work of [63] by replacing the generative model with conditional models called conditional random fields. They show the effectiveness of this approach for the task of noun coreference for natural language processing [53, 54], and also for the task of citation matching [78, 25].

Singla and Domingos [72, 73] also try to address the problem of similar entries, which can not be detected by string distance metrics. They

build up a graph containing all candidate pairs using conditional random fields. Pairs that are clear duplicates, but contain different values in individual attributes, are used to infer that these values are equivalent. This information is then propagated through the graph and used for further duplicate detection. This method can be seen as defining the similarity score for such observed equivalent values to 1, i.e. they are synonyms. Similarly to [8, 7], the naive approach declaring values as synonyms that were used once as equivalent, has the significant drawback that if one value is actually really wrong like "VLDB-94" instead of "VLDB-95" for a conference event, declaring these as synonyms would cause many false matches. Similarly to Domingos, Dong et al. [28] build up a graph and propagate duplicate information to related objects.

Kalashnikov et al. [44] present an interesting approach called "Relationship-based Data Cleaning", which also builds up a graph between all objects connected with edges that are weighted with a confidence and a relevance (called connection strength). Then a global optimal solution is calculated, which determines the most probable duplicate objects. The main drawback is its need to build up a new graph-based data structure and its very computationally expensive calculation procedure.

Similarly, Cohen et al. [21] attempt to find an optimal global solution. They prove that this task is NP-hard and instead propose a priority-queue based algorithm. However, an empirical evaluation, the model calibration and many implementation details are missing, making it hard to compare it to other approaches.

Hill [39] got very promising results in author matching by only analyzing the citation graph of the publication. She calls the used method social network vector-space relational model, which needs prior background knowledge about the social network of the authors and is then able to classify new references to be a specific author of the network.

These graph-based decision models are very interesting and promising approaches, but they still suffer from the complexity problem of building up such a graph-based data structure and the computational problem of calculating a globally optimal solution.

Chapter 3

A Precise Blocking Method

A blocking methods selects a set of candidate duplicates out of the set of all possible pairs. This is necessary to reduce the number of pairs to be examined in the expensive matching phase. The main issue for a blocking algorithm is therefore a small set of resulting candidate duplicates, but additionally a very high duplicate detection ratio and efficiency is important.

The proposed blocking method is a variant of the sorted-neighborhood method as presented by Hernandez-Stolfo [38]. The algorithm for examining a single data source (or already merged data sources) is shown in Figure 3.1.

The records are sorted based on a key value, which can be a single attribute value or the result from a complex key expression. It is preferable to use simple sorting keys, which in practice often allows to use existing indices on the key to get the sorted list of records. The complexity of sorting the records is in general $O(n * logn)$.

The sorted list of records is then sequentially scanned for potential duplicates by comparing all keys within a sliding window. In contrast to Hernandez-Stolfo [38] a dynamically sized window is used, where the size depends on a fixed distance between the key values. The distance between the key values is determined by a distance function like edit-distance. This dynamically sized window is in practice often much smaller than a fixed sized window, but larger when required. Figure 3.2 visualizes the concept of dynamic window sizes.

In the context of data bases sorting is typically done by creating an index on the key value. In this case it is possible to move the window over the key values in the index instead of the records itself. One consequence of this is that the distance of different key pairs is only calculated once, whereas otherwise this costly distance calculation might be done repeatedly or needs

```
function detectDuplicates(records S, key K)
  sort S on K

  init W with ()

  for each key1 in S
    for each key2 in W
      if(distance(key1, key2) < Threshold)
      then potential-duplicates(
             records of key1,
             records of key2
           )
      else remove key2 from W
    add key1 to W
```

Figure 3.1: Blocking Algorithm on Single Source

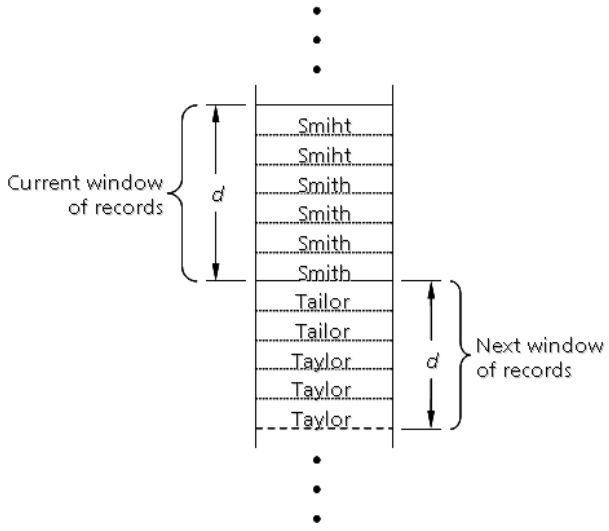

Figure 3.2: The Sorted-Neighborhood Method with Dynamic Window Sizes

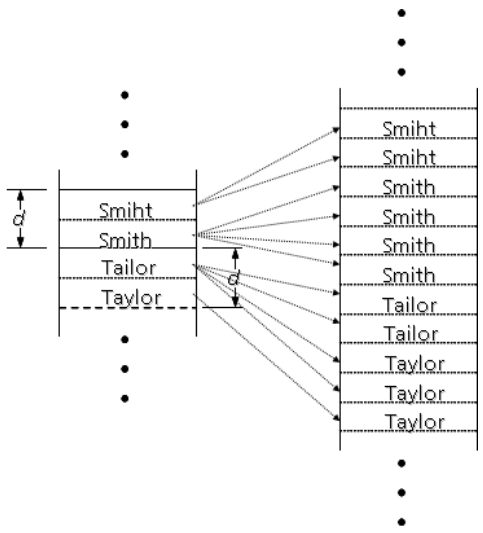

Figure 3.3: Moving the window over the keys instead of the records itself

at least some lookup mechanism (see Figure 3.3).

The complexity of this algorithm is then $O(n * logn + k * w)$, where k is the number of different key values and w is the average size of the window. k depends on the uniqueness of the selected blocking key.

If two data sources or a new data source and a set of already integrated data sources should be examined for duplicates, the algorithm can be slightly modified as shown in Figure 3.4. In this case the new or smaller data source (call it S_2) needs no sorting, but only the larger or integrated data set (call it S_1) is sorted. Then for every key value in S_2 the corresponding position in the sorted set S_1 is determined and all records in the dynamically sized window around this position are used as candidate duplicates.

The complexity of this modified algorithm is $O(n_1 * logn_1 + k_2 * (logn_1 + w))$, where k_2 is the number of different keys in S_2 and w is again the average size of the window. It can be shown that this complexity is always smaller than the complexity of the first algorithm, therefore it should be avoided to combine data sources before duplicate detection. The sorting of the integrated data set is in practice done only once and therefore ignorable in the long run, which further reduces the complexity of the algorithm.

As long as the sorting of the keys is in general not done by distance, but mostly alphabetically, it is clear that not all keys within the distance threshold are also within the window. In other words, for the three sorted keys with $key1 < key2 < key3$ it does not necessarily hold that $dist(key1, key2) <$

```
function detectDuplicates(records S1, records S2, key K)
  sort S1 on K

  for each key1 in S2
    findPos P of key1 in S1

    init PW with P-1
    init key2 with key at PW
    while (distance(key1, key2) < Threshold)
      potential-duplicates(
        records of key1,
        records of key2
      )
      init PW with PW-1
      init key2 with key at PW

    init PW with P+1
    init key2 with key at PW
    while (distance(key1, key2) < Threshold)
      potential-duplicates(
        records of key1,
        records of key2
      )
      init PW with PW+1
      init key2 with key at PW
```

Figure 3.4: Blocking Algorithm on Two Sources

$dist(key1, key3)$. To this end, it is better to do multiple passes with different keys, instead of selecting just a single sorting key for the blocking process. Experiments have shown that such a multi-pass approach provides higher recall and precision for the resulting potential duplicate set, even with much smaller window sizes and therefore fewer comparisons than a single-pass approach. This is consistent with the results of the original sorted-neighborhood method of Hernandez-Stolfo [38].

Several configuration steps must be done for this approach. At first the selection of a good sorting key is important in the same way as for the original approach. A typical selection criterion is the uniqueness of an attribute, i.e., the ratio between the number of different keys to the number of records, because if an attribute is highly unique the window size will be small.

Further a distance function must be selected. This depends on the data or the typical errors on the data as already mentioned in Section 2.2. However, the distance function must also match the sorting method, e.g., for typical alphabetically sorted records only character-based distance measures make sense, whereas numerically sorted records can be compared with an arithmetic difference function. Finally, the threshold or window size must be selected. This depends on the reliability of the key, because for highly reliable keys only a small distance will be regarded as equivalent value.

Chapter 4

Extending the Fellegi-Sunter Model

The Fellegi-Sunter model is a classic approach that is widely used for record linkage (described in Section 2.3.3). It uses the ratio $w(\gamma) = m(\gamma)/u(\gamma)$ as decision function. The problems using this function are in general the following: 1. how to find a threshold on $w(\gamma)$; 2. how to determine the parameters $m(\gamma)$ and $u(\gamma)$, when using continuous similarity measures and taking dependencies into account.

4.1 A Probability Interpretation

For the duplicate decision, every application needs to set a threshold on the duplicate estimation value $w(\gamma)$. A natural threshold for this ratio is 1, because a value greater than 1 means that the probability $m(\gamma)$ is higher than the probability $u(\gamma)$. However, this ratio ignores the general expectation how many duplicates exist in this data set, and values other than 1 are difficult to interpret as they do not scale linearly.

In order to include a general expectation for duplicates and to make the result more intuitively interpretable, a probability interpretation for the Fellegi-Sunter model is defined as the conditional probability of a and b being duplicates (element of M), given the comparison vector γ. This conditional probability can be calculated as follows:

$$P((a,b) \in M \mid \gamma) = \frac{m(\gamma) * P(M)}{m(\gamma) * P(M) + u(\gamma) * P(U)} \tag{4.1}$$

This formula follows directly from the Bayes rule [67] and the total probability theorem:

$$P((a,b) \in M \mid \gamma) = \frac{m(\gamma) * P(M)}{P(\gamma)} \tag{4.2}$$

$$P(\gamma) = m(\gamma) * P(M) + u(\gamma) * P(U) \tag{4.3}$$

The probability $P(M)$ is the prior probability that two records are duplicates and represents the general expectation for duplicates. It is defined as the ratio between the set of duplicates and the set of all pairs:

$$P(M) = \frac{|M|}{|M| + |U|} \tag{4.4}$$

$P(U)$ is simply the complement of $P(M)$; $P(U) = 1 - P(M)$.

The main difference to the original Fellegi-Sunter ratio is that now the values are linearly scaled in the range {0,1}, whereas in Fellegi-Sunter the range is {0,infinite}. The introduced term $P(M)$ includes the general expectation for duplicates, i.e., if there are only few duplicates expected, $P(M)$ will be small and accordingly the matching probability for potential duplicates will be small as well. The natural threshold is now 0.5, i.e., a pair is naturally declared as duplicate, when the probability is greater than 50%. This is identical to the natural threshold of 1 in the original Fellgi-Sunter ratio, in the case that $P(M)$ equals 0.5.

However, the order based on the original Fellegi-Sunter ratio and the order based on the Bayes formula 4.1, is always identical. This can be easily shown, by transforming the inequation based on Bayes (4.5) into the equivalent inequation based on the original Fellegi-Sunter ratio (4.6):

$$\frac{m(\gamma_1) * P(M)}{m(\gamma_1) * P(M) + u(\gamma_1) * P(U)} < \frac{m(\gamma_2) * P(M)}{m(\gamma_2) * P(M) + u(\gamma_2) * P(U)} \tag{4.5}$$

$$\Longleftrightarrow$$

$$\frac{m(\gamma_1)}{u(\gamma_1)} < \frac{m(\gamma_2)}{u(\gamma_2)} \tag{4.6}$$

4.2 Estimation of the $m(\gamma)$ Parameter

When assuming independence between the attributes and mapping the continuous similarity values of the comparison vector to boolean values using some threshold, the individual probability $m_i(\gamma_i[a,b])$ for an individual similarity value of an attribute-pair $(\gamma_i[a,b])$ is the ratio between the number of all pairs in M that match with this $\gamma_i[a,b]$ to the size of M.

$$m_i(\gamma_i[a,b]) = \frac{|\{(x,y) \in M \mid \gamma_i[x,y] = \gamma_i[a,b]\}|}{|M|} \tag{4.7}$$

The overall $m(\gamma[a,b])$ is simply the product of the individual probabilities:

$$m(\gamma[a,b]) = \prod_{i}^{n} m_i(\gamma_i[a,b]) \tag{4.8}$$

When taking dependencies between the attributes into account, the probability $m(\gamma[a,b])$ can be estimated by the ratio between the number of all pairs in M, where *all* individual components of $\gamma[a,b]$ match, to the size of M.

$$m(\gamma[a,b]) = \frac{|\{(x,y) \in M \mid \forall\gamma_i[x,y] = \gamma_i[a,b]\}|}{|M|} \tag{4.9}$$

In the case when the continuous similarity values should not be mapped to boolean values the estimation of $m(\gamma[a,b])$ is not so obvious. But using continuous values is definitely preferable, because value-pairs are not always clearly categorizable as definite match or not match, but there is a whole range of similarity, which makes it hard to define a good threshold.

In order to define the probability $m(\gamma[a,b])$ for the continuous case, the following assumption is made:

Assumption 4.1 *The probability* $m(\gamma)$ *is monotonically* increasing *with the* increase *of the similarity between value-pairs, i.e., if the similarity measures in* $\gamma[a,b]$ *are greater than the similarity measures in* $\gamma[x,y]$, $m(\gamma[a,b])$ *is greater than* $m(\gamma[x,y])$.

Using this assumption the independent probability $m_i(\gamma_i[a,b])$ can be defined as the ratio between the number of all pairs $[x,y]$ in M whose $\gamma_i[x,y]$ is less than or equal to $\gamma_i[a,b]$, to the number of all pairs in M. Without the independence assumption this corresponds to the number of all pairs in

M whose comparison vector is absolutely less than $\gamma[a,b]$ to the number of all pairs in M.

$$m_i(\gamma_i[a,b]) = \frac{|\{(x,y) \in M \mid \gamma_i[x,y] <= \gamma_i[a,b]\}|}{|M|} \tag{4.10}$$

$$m(\gamma[a,b]) = \frac{|\{(x,y) \in M \mid \forall\gamma_i[x,y] <= \gamma_i[a,b]\}|}{|M|} \tag{4.11}$$

4.3 Estimation of the $u(\gamma)$ Parameter

$u(\gamma[a,b])$ is analogous to the definition for $m(\gamma[a,b])$. In the most simple case assuming independence between the attributes and mapping the continuous distance values of the comparison vector to boolean values using some threshold, the individual probability $u_i(\gamma_i[a,b])$ for an individual similarity value of an attribute-pair $(\gamma_i[a,b])$ is the ratio between the number of all pairs in U that match with this $\gamma_i[a,b]$ to the size of U.

$$u_i(\gamma_i[a,b]) = \frac{|\{(x,y) \in U \mid \gamma_i[x,y] = \gamma_i[a,b]\}|}{|U|} \tag{4.12}$$

The overall $u(\gamma[a,b])$ is simply the product of the individual probabilities:

$$u(\gamma[a,b]) = \prod_i^n u_i(\gamma_i[a,b]) \tag{4.13}$$

Without the independence assumption between the attributes, the probability $u(\gamma[a,b])$ can be defined as the ratio between the number of all pairs in U, where all individual components of $\gamma[a,b]$ match, to the size of U.

$$u(\gamma[a,b]) = \frac{|\{(x,y) \in U \mid \forall\gamma_i[x,y] = \gamma_i[a,b]\}|}{|U|} \tag{4.14}$$

In order to define the probability $u(\gamma[a,b])$ for the continuous case, the following assumption is made:

Assumption 4.2 *The probability $u(\gamma)$ is monotonically* increasing *with the* decrease *of the similarity between value-pairs, i.e., if the similarity measures in $\gamma[a,b]$ are less than the similarity measures in $\gamma[x,y]$, $u(\gamma[a,b])$ is greater than $u(\gamma[x,y])$.*

Using this assumption the independent probability $u_i(\gamma_i[a,b])$ can be defined to be the ratio between the number of all pairs in U whose γ_i is greater than or equal to $\gamma_i[a,b]$, to the number of all pairs in U. Without the independence assumption this corresponds to the number of all pairs in U'_t whose comparison vector is absolutely greater than $\gamma[a,b]$ to the number of all pairs in U.

$$u_i(\gamma_i[a,b]) = \frac{|\{(x,y) \in U \mid \gamma_i[x,y] >= \gamma_i[a,b]\}|}{|U|} \tag{4.15}$$

$$u(\gamma[a,b]) = \frac{|\{(x,y) \in U \mid \forall\gamma_i[x,y] >= \gamma_i[a,b]\}|}{|U|} \tag{4.16}$$

Chapter 5

Unsupervised Matching

This chapter presents the developed approach for the matching phase. The first section explains the basic architecture, the following sections then present the methods and algorithms of the individual modules in detail.

5.1 Overview

The architecture consists of the usual two phases that are blocking, which generates a set of candidate duplicates, and matching, which compares the candidate duplicates in detail and makes the final duplicate decision. However, the matching phase in the approach takes not only a set of candidate duplicates (M') as input, but additionally a set of non-duplicates (U'). This additional input should enable the matching algorithm to work completely unsupervised.

The idea is that the matching algorithm is able to remove the non-duplicates from M' given the set of sample true non-duplicates of U'. Therefore these sample non-duplicates must be representative for the non-duplicates in M', which are some kind of similar object-pairs, as otherwise they would not have been declared as candidate duplicates.

For a large family of applications this set of representative non-duplicates can be generated in an unsupervised way. These applications are characterized by the following properties:

- only duplicates between different data sources need to be detected
- these individual data sources themselves are more or less duplicate free

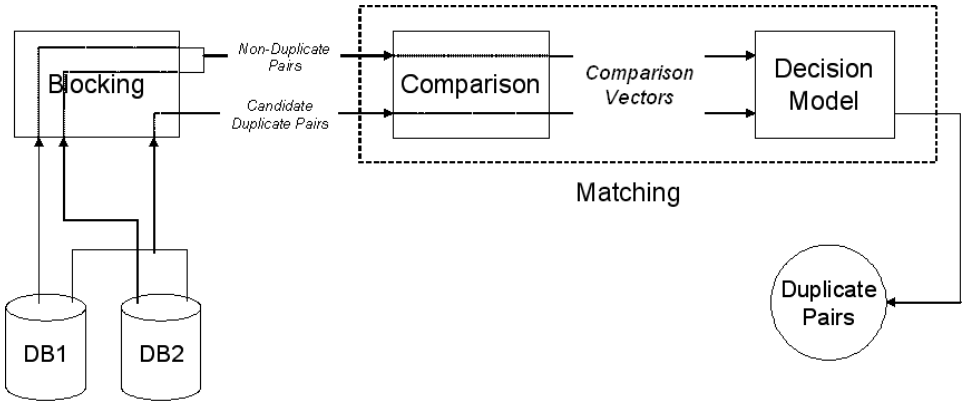

Figure 5.1: Architecture of the duplicate detection system

These high quality data sources typically occur in commercial settings like webshops or product catalogs or other manually maintained data sources, e.g., the publication data bases DBLP [51] and CompuScience [31].

When these properties hold for the given application and data sources, a representative set of non-duplicates U' can be generated by applying the same blocking algorithm to the sets of object-pairs within the individual data sources ($A \times A \cup B \times B$). This set shows the same kind of similarity as the non-duplicates in M' ($A \times B$), because they were generated using the same algorithm. This architecture is shown in Figure 5.1.

If the properties do not hold for the given application, because duplicates within a single data source should be detected or the individual data sources contain too many duplicates, the set of non-duplicates U' must be generated differently, e.g., by manually selecting such pairs. In the following, this thesis assumes that these properties hold.

The set of candidate duplicates M' is generated as usual by blocking on the set of object-pairs between the data sources ($A \times B$). It consists of the set of actual duplicates M and additional non-duplicates U.

After the sets M' and U' are generated by the blocking module, a comparison module compares the given object-pairs in detail and produces comparison vectors for all pairs in M' and U'. Given these comparison vectors as input, the decision module tries to identify actual non-duplicates in M' based on the samples from U' and finally generates a set of duplicate pairs. This remaining set of duplicates should be ranked by some kind of confidence into this decision.

The following Section 5.2 defines how the comparison module works.

Section 5.3.1 show how the Fellegi-Sunter model and Section 5.3.2 and 5.3.3 show how machine learning algorithms can be used as decision models in such a scenario.

5.2 The Comparison Module

The comparison module compares object-pairs in detail and creates a comparison vector γ for every such object-pair. An individual component of this comparison vector is a similarity measure for an individual attribute-pair. Such a similarity measure is defined to be greater if more similarity is detected, the range for such a similarity value can be {-infinite,infinite}. One typical kind of similarity measures are distance functions, which are dissimilarity measures with a range of {0,1}, where 0 means identical and 1 stands for maximal distance. These distance values are translated to a similarity measure by simply taking the complement of it.

The influence on the overall matching performance of such similarity measures can be enormous, i.e., a decision module has only a chance if the distribution of the similarity values between the duplicates and the non-duplicates are significantly different. But the selection of an appropriate similarity measure is difficult as several kinds of data errors are possible as already mentioned in Section 2.2.

Therefore the comparison module allows to use several similarity measures for the same attribute-pair, like a string-edit-distance and an arithmetic difference for a "year" attribute, and stores the individual similarity values in individual comparison vector components. That means the length of a comparison vector may not be identical with the number of attribute-pairs. Note that then generally the independence assumption does not hold.

Null Values Datasets often contain optional attributes, i.e., attributes may contain null values. E.g., in the person domain, the middle name of a person often either does not exist or is not available. In general such null values cannot be used by similarity functions.

Therefore the comparison module allows a comparison vector component to be either a continuous similarity value, or "null", indicating that this similarity function was not able to calculate a similarity score for this attribute-pair. How a null value is treated for the duplicate decision is the task of the decision module.

$$\gamma_i \in \mathbb{R} \cup \text{"null"} \tag{5.1}$$

Multi-valued Attributes So far only flat records are handled with the approach. In order to extend it to multi-valued attributes, i.e., attributes that contain an arbitrary number of individual values, e.g., the list of authors of a publication, the approach must be slightly modified.

Comparing multi-valued attributes requires comparing all corresponding individual values and results in a sequence of similarity values, i.e., a comparison vector itself. If one multi-valued attribute contains less individual values than the other, this is seen as missing values, i.e., containing null values. The way of finding the corresponding value depends on whether the values are ordered or not. If the multi-valued attributes are ordered, e.g., the author names of a publication, simply the values at the same position are compared. Unordered multi-valued attributes require some form of set comparison, selecting the comparison vector with the shortest length.

Using a comparison vector with arbitrary length n_i directly as value within the comparison vector (γ) of the object-pair to be examined, would result in independent distributions for every occurring length of this multi-valued attribute. Therefore it is preferable to collapse the comparison vector of the multi-valued attribute to a single similarity measure.

In order to collapse such a comparison vector, the following assumptions are made:

- the relevance of every individual value is the same, i.e., the similarity of every individual value contributes the same amount to the overall similarity of the set.
- the similarities between the individual values are independent, i.e., there is no dependency between the individual values.

Under these assumptions the individual similarity values γ_{ik} can be collapsed to a single similarity value by simply using the arithmetic mean of the individual similarity values. If several similarity functions were used for the comparison of the attribute-pairs, the mean of every such similarity function must be calculated.

$$\gamma_i = \frac{\sum_k^{n_i} \gamma_{ik}}{n_i} \tag{5.2}$$

Relationships Beside multi-valued attributes in many data models also relationships to other objects are possible. An example of such a relationship is the *conference paper* to its *conference*. When supporting relationships in

the approach, it is necessary to compare the related objects, i.e., the conferences. This is done in an independent duplicate detection process, which needs its own sets of candidate duplicate conferences (M'_r) and representative non-duplicate conferences (U'_r). In order to obtain these input sets, a blocking phase is not necessary, but these sets can be obtained from the initial candidate duplicate conference papers M' and non-duplicate conference papers U'.

To this end, all distinct pairs of conferences from M' form M'_r and all distinct pairs of conferences from U' form U'_r. This initial set U'_r may still contain duplicates, because although U' only contains clear non-duplicates, the related objects may still be, e.g., non-duplicates conference papers may still be published at the same conference. However, because U' contains only pairs from the same source, duplicate related objects must be identical and therefore directly identifiable as such. That means U'_r can be easily filtered to contain only non-duplicates.

Using these sets M'_r ande U'_r as input for a separate duplicate decision, all related object-pairs get some kind of duplicate estimation score, which can then be used as similarity value in the comparison vector for the original duplicate detection process.

However, relationships offer additional information that can be used for their duplicate estimation. This is the observed frequency $o(a_r, b_r)$ that two related objects a_r and b_r occur together within candidate duplicates. If this frequency is significantly higher than the randomly expected frequency $e(a_r, b_r)$, this strongly indicates duplicate related objects even if their similarity score calculated by convential methods is low.

$$o(a_r, b_r) = \frac{|\{(a_r, b_r) \in M\}|}{|M|} \tag{5.3}$$

$$e(a_r, b_r) = \frac{|\{(a_r, .) \in M\}|}{|M|} * \frac{|\{(., b_r) \in M\}|}{|M|} \tag{5.4}$$

To this end an additional kind of similarity measures is suggested in the case of relationships using this information. Such similarity measures are known as association measures [30]. The most commonly used association measures in information theory are the *mutual information score (mi)* [19] and the *t-score* [18].

$$mi = \log \frac{o(a_r, b_r)}{e(a_r, b_r)} \tag{5.5}$$

$$t - score = \frac{o(a_r, b_r) - e(a_r, b_r)}{\sqrt{o(a_r, b_r)}} \tag{5.6}$$

Although the *mi* measure is particularly prone to overestimate low-frequency data (where $e(a_r, b_r)$ is small), it has nonetheless become a de facto standard in (mainly British) lexicography. From a theoretical perspective, the *t-score* is difficult to motivate, but surprisingly, it has shown to perform quite well in collocation extraction tasks and is often used together with the *mi* measure.

It should be noted that not only explicitly modeled relationships with objects each consisting of several attributes can profit from such measures, but also single non-unique attributes in flat record lists (unnormalized relations), e.g., the city and cuisine attribute in the restaurant example (see Table 1.1).

If relationships are cyclic, e.g., if the conference has an attribute *papers* relating to its conference papers, then following such a relationship recursively would result in an infinite loop. Therefore relationships are ignored that refer to objects, which are currently estimated, i.e., the attribute *papers*.

5.3 Decision Models

The decision module is faced with the problem to find a way to separate the data points represented by their comparison vectors into the two classes duplicates and non-duplicates. Figure 5.2 shows a sample distribution of such data points, where the circles represent duplicates and the squares represent non-duplicates.

Several decision models can be used for this task. This thesis focus on the use of statistic methods, i.e., the extended Fellegi-Sunter model and machine learning methods, i.e., KMeans and Support Vector Machines.

5.3.1 The Extended Fellegi-Sunter Model

The extended Fellegi-Sunter model as described in Chapter 4 can be used as decision model given only the input sets M' and U'. In this case only an approximation for $P((a, b) \in M \mid \gamma[a, b])$ can be calculated that is called $P'((a, b) \in M \mid \gamma[a, b])$ based on approximations of $m'(\gamma[a, b])$ and $u'(\gamma[a, b])$, which are introduced below:

$$P'((a, b) \in M \mid \gamma[a, b]) = \frac{m'(\gamma[a, b]) * P'(M)}{m'(\gamma[a, b]) * P'(M) + u'(\gamma[a, b]) * P'(U)} \tag{5.7}$$

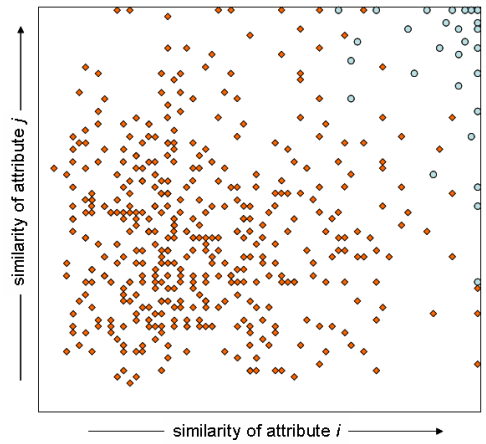

Figure 5.2: Sample Distribution of Duplicates (circles) and Non-Duplicates (squares)

Estimation of $m'(\gamma)$ For estimating $m'(\gamma[a,b])$ and $u'(\gamma[a,b])$ simply the sets M' and U' are used. In particular, the parameter $m'(\gamma[a,b])$ for continuous values in $\gamma[a,b]$ and taking dependencies into account, is defined as the ratio of the number of all pairs in M' whose comparison vector is absolutely less than $\gamma[a,b]$ to the number of all pairs in M'.

$$m'(\gamma[a,b]) = \frac{|\{(x,y) \in M' \mid \forall \gamma_i[x,y] <= \gamma_i[a,b]\}|}{|M'|} \tag{5.8}$$

This $m'(\gamma[a,b])$ is only an approximation of the true $m(\gamma[a,b])$ as it also counts the non-duplicates in M'. In general this value will be greater as the true value, but this can not be guaranteed, because the denominator is also greater. It is also possible to remove detected non-duplicates iteratively from M' and recalculate the $m'(\gamma[a,b])$ parameter in every iteration. This allows to get better $m'(\gamma[a,b])$ values and with this better $P'((a,b) \in M \mid \gamma[a,b])$ values, but experiments show that this does generally not increase the overall duplicate detection accuracy.

For boolean values in $\gamma[a,b]$ or under the independence assumption the definition is analogous to the definition of $m(\gamma[a,b])$ in Section 4.2.

Estimation of $u'(\gamma)$ Similarly, the parameter $u'(\gamma[a,b])$ for continuous values in $\gamma[a,b]$ and taking dependencies into account, is defined as the ratio of the number of all pairs in U' whose comparison vector is absolutely greater than $\gamma[a,b]$ to the number of all pairs in U'.

$$u'(\gamma[a,b]) = \frac{|\{(x,y) \in U' \mid \forall \gamma_i[x,y] >= \gamma_i[a,b]\}|}{|U'|} \tag{5.9}$$

This $u'(\gamma[a,b])$ is also only an approximation of the true $u(\gamma[a,b])$ as it is calculated on the set U', which has no direct relation to U. That means, this approximation is only valid, if the non-duplicates in U' are actually representative for the non-duplicates in U. For boolean values in $\gamma[a,b]$ or under the independence assumption the definition is analogous to the definition of $u(\gamma[a,b])$ in Section 4.3.

Estimation of $P'(M)$ $P'(M)$ is the prior probability that some pair in M' is actually a duplicate. It can be estimated by the ratio of pairs in M', which have a probability of being a duplicate $P'((a,b) \in M \mid \gamma[a,b])$ higher than 50% to all pairs in M'. Because this probability $P'((a,b) \in M \mid \gamma[a,b])$ already requires the parameter $P'(M)$, this can only be calculated iteratively:

$$P'_t(M) = \frac{|\{(a,b) \in M' \mid P'_{t-1}((a,b) \in M \mid \gamma[a,b]) >= 0.5\}|}{|M'|} \tag{5.10}$$

The initial $P'_0(M)$ must be greater than 0 and less than 1. Typically it is set to 0.5. This iteration always converges, because the definitions of $P'(M)$ and $P'((a,b) \in M \mid \gamma[a,b])$ have a close relationship, i.e., if $P'(M)$ is decreasing than also $P'((a,b) \in M \mid \gamma[a,b])$ is decreasing and if $P'((a,b) \in M \mid \gamma[a,b])$ is decreasing than also $P'(M)$ is getting less or equal. Experiments also show that this $P'(M)$ is a good approximation for the true $P(M)$.

Handling Null Values The problem of handling null values occurs for the determination of $u'(\gamma)$ and $m'(\gamma)$ during the comparison of a $\gamma_i[x,y]$ and $\gamma_i[a,b]$, when one of these values is the null value and the other is a continuous similarity value. There are several ways to decide, if such components match or not:

- A null value never matches a distance value, i.e., a comparison vector containing null values can only be compared with comparison vectors that also contain null values for exactly the same component. To this end, the sets U' and M' are split into subsets, one for each null value combination. These subsets might be very small depending on the size

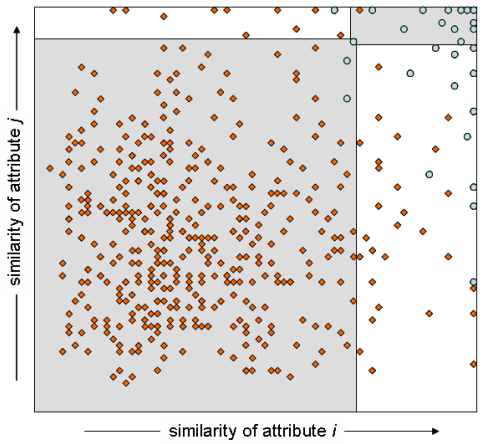

Figure 5.3: Concept of the Fellegi-Sunter Decision Model

of the input sets and this would result in poor accuracy/confidence for the probabilities.

- A null value matches every distance value. This basically means that this comparison vector component is ignored and has no further influence on the resulting probabilities.
- The null value is replaced with a similarity value. This similarity value can be the most probable value for this comparison vector component, which is e.g. the mean value of all non-null similarity values for this component.

Conceptual View Conceptually the Fellegi-Sunter based approach can be visualized by Figure 5.3, for a specific data point (i.e. object-pair) it counts the number of absolutely smaller duplicates (in the lower left corner) and the number of absolutely greater non-duplicates (in the upper right corner).

5.3.2 Support Vector Machines

The problem can also be seen as classification problem, where the unlabeled data points in M' should be classified into the two distinct classes of duplicates M and non-duplicates U. Classification algorithms are in general supervised methods, which use a set of labeled examples to train a classification function, which is then used to classify the unlabeled data points.

State of the art for supervised classification are Support Vector Machines (SVM) [67].

Support Vector Machines try to find a hyperplane between the two classes, such that the margin between the hyperplane and the data points of both classes is maximized. SVMs use kernel functions that are able to separate many kinds of cluster shapes. SVMs need to be trained with data points of both classes. However, the problem at hand has additional labeled data points for only one class, which are the data points in U' (further called negative examples E_n).

In order to obtain good labeled data points for duplicates (further called positive examples E_p), two methods are possible. First the pairs in the set M' itself can be used as duplicate training examples. This results in poor precision initially, but still some pairs from M' are classified as non-duplicates. This allows to retrain the SVM with a revised set of positive examples E_{p1} that only contains those pairs that were classified as duplicates by the initial classification. This can be iteratively done until no more pairs in E_{pt} are classified as non-duplicates:

$$E_{pt} = \{\gamma \in M' \mid svm_{t-1}(\gamma) >= 0\} \tag{5.11}$$

Experiments show that this works well, if the ratio of non-duplicates to duplicates in M' is not too high. However, otherwise the SVM approach, even with iteration, might completely fail, i.e., it results in a very poor precision. Therefore another method to create the initial set E_{p0} is to use the extended Fellegi-Sunter model for this:

$$E_{p0} = \{(a,b) \in M' \mid P'((a,b) \in M \mid \gamma) >= 0.5\} \tag{5.12}$$

SVMs are parameterized and these parameters must in general be learned from the training set. However, as no clean training sets exist, no way has been found to learn these parameters. Training the parameters using the set M' or the already filtered set from the Fellegi-Sunter model as positive examples have resulted in significantly worse accuracy. Therefore default parameter settings are used in this approach, i.e., 1 for the cost and the gamma parameter using the radial basis kernel. This nevertheless shows very good results.

Figure 5.4 shows the basic concept of the SVM model, it basically searches for a separation line between the given duplicates and non-duplicates, whose margin to all data points is maximal.

Null values are handled by replacing them with the most probable similarity value for this component, which is the mean of all non-null similarity

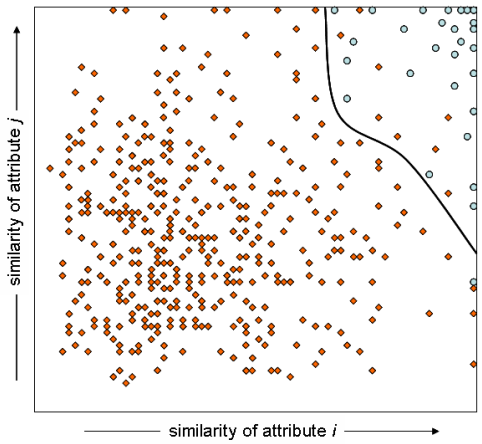

Figure 5.4: Concept of the SVM Decision Model

values. Recently, also SVMs that use more sophisticated methods for handling null values are proposed [64].

5.3.3 KMeans

The problem can further be seen as clustering problem, where the unlabeled data points in M' should be clustered into two distinct classes of duplicates M and non-duplicates U. Clustering algorithms are in general unsupervised methods, which try to separate the unlabeled data points into two clusters of data points that seem to belong together based on some distance between the data points. Examples for such clustering algorithms are KMeans [52] and spectral clustering [71].

The KMeans algorithm tries to find two cluster centroids, such that all data points in this cluster are nearer to their centroid than to the other. To this end KMeans starts with initial centroids, assigns all data points to the centroid that is nearest to them and afterwards calculates new centroids for the identified clusters. This is done iteratively until the centroids stay fixed. As the distances to the centroid are in general calculated by a simple euclidean distance, the assumed cluster shape is spherical.

This is the main disadvantage of KMeans: it fails, if the cluster shape strongly differs from a spherical shape. The features of the comparison vector in duplicate detection usually show very different distributions (based on the uniqueness and reliability of the attribute) and they may also depend on each other, which may result in non-spherical cluster shapes, like in the

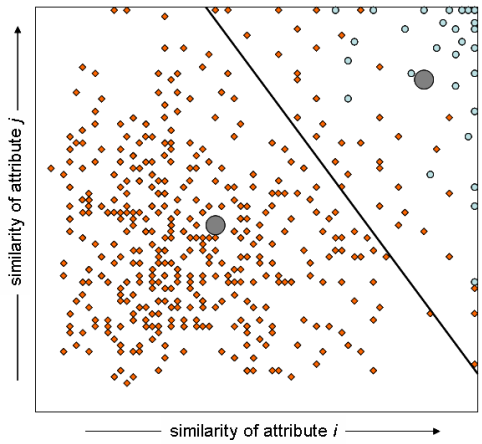

Figure 5.5: Concept of the KMeans Decision Model

sample distribution in Figure 5.2.

In order to use KMeans, one has to specify the initial cluster centroids. In the given scenario this can be achieved by calculating the centroid for the non-duplicates out of U' and setting the centroid of the duplicates to 0. However, the results of the experiments are very poor, which matches the expectation that KMeans is not well suited for this kind of application.

Figure 5.5 shows the basic concept of the KMeans model, it searches two centroids (visualized as large grey circles) and assigns all data points to the nearest centroid, which results in a linear separation.

5.3.4 Optimization for Large Data Sets

The complexity in the Fellegi-Sunter model to calculate $m'(\gamma)$ and $u'(\gamma)$ is between $O(n+m)$ for a naive counting and $O(logn+logm)$ using some kind of caching or indexing optimization, where n is the size of M' and m is the size of U'. This calculation must be done for every pair in M', therefore the complexity of the decision is between $O(n*(n+m))$ and $O(n*(logn+logm))$. The complexity to train the SVM is also quadratic with the size of M' and U'. This requires an optimization in the case that M' and U' are large. Fortunately, a simple random sampling on these sets produces very good results and allows linear complexity of the decision process. The number of required samples as shown in the experiments is less than 1000.

Chapter 6

Evaluation

This chapter presents the conducted experiments with their used settings and their results. The experiments are conducted on several different data sets, i.e., data sets that were previously used as benchmarks in other related work, and large data sets that show the usability and scalability of the approaches also on real world examples. These data sets are presented in detail in the following Section 6.1.

The experiments in Section 6.2 evaluate the blocking approach by comparing it to the original sorted-neighborhood method and to other state of the art methods. Thereby the resulting accuracy of the candidate duplicates as well as the costs for reaching these results are compared. Furthermore, the scalability is evaluated on differently sized data sets.

The experiments in Section 6.3 evaluate the matching approach by comparing the extended Fellegi-Sunter model taking dependencies into account and using continuous comparison values to the original Fellegi-Sunter model. Then, the machine learning methods are used as decision models and these results are compared to the Fellegi-Sunter approach, to simple baseline methods as well as to other existing unsupervised and supervised approaches. Thereby the accuracy in terms of precision and recall of the different approaches is evaluated as usual in information retrieval. In addition, again, the scalability of the Fellegi-Sunter and the support vector machine approach is evaluated.

Finally, the impact on the final duplicate detection result using different similarity measures in the comparison module is evaluated.

Table 6.1: Sample duplicate records from the *Restaurant* data set

name	address	city	cuisine
uncle nick's	747 ninth ave.	new york city	greek
uncle nick's	747 9th ave. between 50th and 51st sts.	new york	mediterranean

Table 6.2: Sample duplicate records from the *Census* data set

last name	first name	house number	street
JIMENCZ	WILLPAMINA	S 214	BANK
JIMENEZ	WILHEMENIA	214	BANKS

6.1 Data Sets

For the evaluation at first a *Restaurant* and a *Census* data set are chosen, which were previously used as benchmarks for duplicate detection, e.g. in [10, 65]. The restaurant data set contains 864 restaurant names and addresses with 112 duplicates, composed of 533 and 331 restaurants assembled from Fodor's and Zagat's restaurant guides. These individual data sets are duplicate free, the attributes being restaurant name, street address, city and cuisine. Table 6.1 shows a sample duplicate record from this data set.

The census data set is a synthetic data set containing 824 census-like records with 327 duplicates, composed of two duplicate free sets with 449 and 375 records, the attributes being last name, first name, house number and street. Table 6.2 shows a sample duplicate record from this data set.

For the evaluation of the blocking an additional *Mailing* data set is used, which has been previously used as benchmark for blocking methods in [5]. The mailing data set is generated by the DBGen [38] database generator. It generates artificial address records and randomly introduces duplicates with errors. The same settings as [5] are used to generate data sets with approximately 1000, 2000, 5000 and 10000 records, where the number of clusters are half the number of records, which results in most records having a single duplicate. However some records will have more than one other record with a true match and some will have no duplicates. This data set can not be easily separated into individual duplicate free data sets and is therefore not used for evaluating the matching approach. Table 6.3 shows a sample duplicate record from this data set.

Finally a large real world data set from the *Publication* domain is used. This is the DBLP computer science bibliography [51] and the CompuScience

Table 6.3: Sample duplicate records from the *Mailing* data set

last name	first name	street number	street	city
Swenberg	Gruemnkranz	436	Klich Avenue	Anchor Point
Swenbearg	Gruenkranz	436	Klich Avenue	sAnchor Point

Table 6.4: Sample duplicate records from the *Publication* data set

title	journal	volume	year
Representing and reasoning on conceptual queries over image databases.	J. Intell. Inf. Syst. and Database Technologies	v. 14	2000
Representing and Reasoning on Conceptual Queries Over Image Databases.	J. Intell. Inf. Syst.	14	2000

bibliographic database [31]. The versions used, contain 577252 publications for DBLP and 392112 publications in CompuScience, which consist of journal articles, conference articles, conference proceedings, thesis and others. These data sources were provided in an XML format with elements similar to bibtex attributes, depending on the publication type. However, all publication types have at least a title attribute, which is used for evaluating the blocking algorithm. For evaluating the matching approach a subset of both data sets is used, containing all journal articles in the years 1999 to 2002, i.e., 62403 articles in DBLP and 56340 in CompuScience. All candidate duplicates on this subsets based on blocking on the title attribute (21142 potential duplicates), are manually labeled resulting in 19978 true duplicates, which are used to measure precision and recall of the matching algorithm. Table 6.4 shows a sample duplicate record from this data set.

All data sets are transformed into an XML format and stored in an XML database. Indices on every single attribute were created inside the database, which corresponds to a sorting on every attribute.

6.2 Blocking Experiments

This section presents the results of the experiments of the blocking approach. After introducing the relevant measures in the following section, the approach is compared to the original sorted-neighborhood method. Furthermore, it is compared to other state of the art methods and finally, in

order to evaluate the scalability of the approach, it is used on a large real world data set.

6.2.1 Experimental Methodology

For comparison of the experimental results with [5], the "reduction ratio" (RR), "pairs completeness" (PC) and "F score" as defined there, are used instead of the usual precision, recall and F-measures as used in information retrieval [3]. These measures are used, because of the different requirements of the blocking phase, where not the resulting accuracy is most relevant, but a good blocking method finds basically all true duplicates (PC) and results in a small set of candidates (RR).

$$RR = 1 - \frac{|CandidateDuplicates|}{|AllPairs|}$$

$$PC = \frac{|CorrectlyIdentifiedDuplicates|}{|TrueDuplicates|}$$

$$Fscore = \frac{2 * RR * PC}{RR + PC}$$

The pairs completeness is equivalent to the definition of recall, whereas the reduction ratio is the relative reduction in the number of candidate duplicates to the number of all possible pairs. The F score is simply the harmonic mean between the pairs completeness and reduction ratio, similar as the F-measure is the harmonic mean between precision and recall. Instead of precision-recall curves, RR-PC curves are presented for a corresponding graphical overview.

6.2.2 Comparison with the Original Method

At first the method is compared to the original sorted-neighborhood method [38] in terms of accuracy and costs.

Accuracy To this end the RR and PC values are measured for several window sizes. Larger window sizes increase the pairs completeness, but decrease the reduction ratio. In this experiment window sizes between 0 and 10 records in single steps are used for the original fixed sized method and windows with distances between 0 and 0.5 in steps of size 0.05 are used

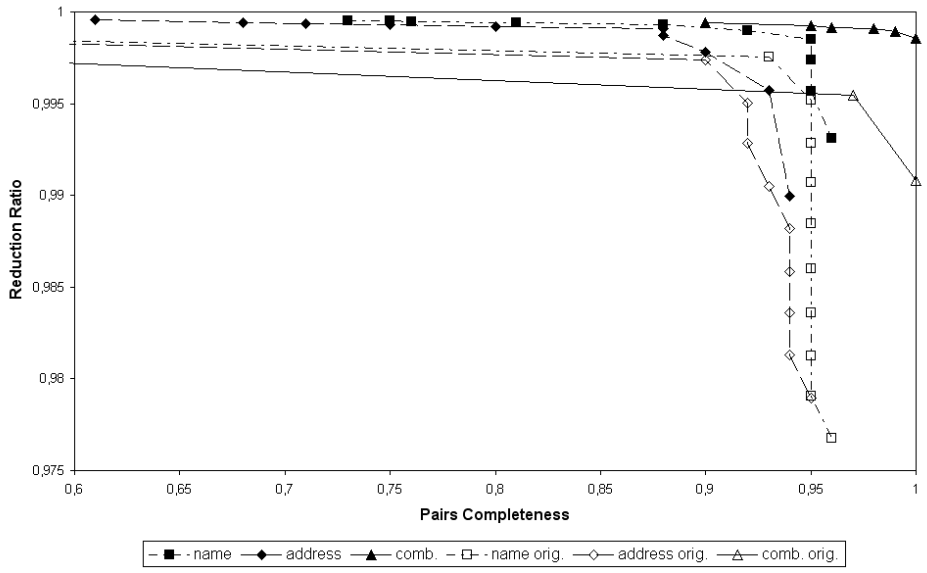

Figure 6.1: RR-PC for the Restaurant data set

for the dynamically sized method. Such a distance value represents the distance as calculated by the Jaro distance measure [22] between the key value of the first and the last record in the window.

Both methods are applied in a single pass on the two individual attributes (name, address) as well as in two passes on both attributes (comb). The RR-PC curve for the restaurant data set can be seen in Figure 6.1, the RR-PC curve for the census data set in Figure 6.2. Every data point represents a window size, smaller window sizes correspond to smaller pairs completeness. For easier comparison, the curves are pruned after the maximum pairs completeness is reached.

Both figures show that the method using dynamic window sizes significantly outperforms the original method in all experiments; i.e., it reaches a higher reduction ratio for the same pairs completeness. It can be further seen that the combined multi-pass approach reaches a much higher accuracy than the single-pass approaches, which matches the results of the experiments in the original approach in [38].

The most interesting point in the curves is where the pairs completeness is maximum, this is a pairs completeness of 1 for the restaurant data set and 0.97 for the census data set. This represents the settings, where most duplicates are detected, which has the highest priority in blocking as not

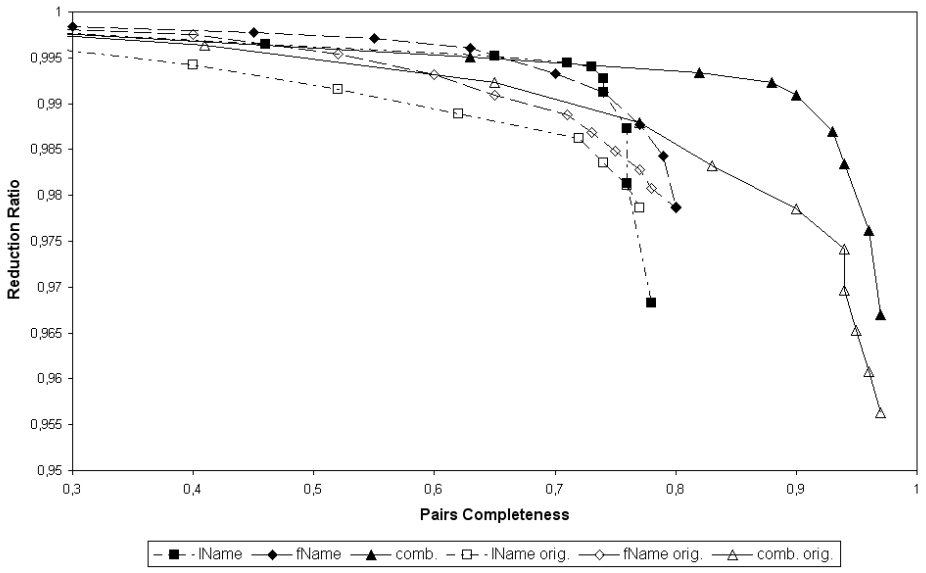

Figure 6.2: RR-PC for the Census data set

detected duplicates during blocking can not be detected in the following phases.

At this point of maximum pairs completeness and for the combined approaches the restaurant data set shows a reduction ratio of 0.999 for this vs. 0.991 for the original approach (i.e., 251 vs. 1622 candidate duplicates); and on the census data set the approach reaches 0.967 vs. 0.956 for the original approach (i.e., 5558 vs. 7367 candidate duplicates).

Costs In order to compare the costs of the original method to the developed approach, the number of distance calculations between individual attributes are measured. The approach needs to calculate such distances during blocking for determining the window, but at the same time it is able to save this distance in the comparison vector of the candidate duplicates for the matching phase. In contrast, the original approach needs no such comparisons during the blocking phase, but has to determine the complete comparison vector distances in the matching phase. Therefore a valid cost comparison is to count the number of distance calculations during the blocking and the matching phase. Figure 6.3 shows the number of calculations in comparison to the reached pairs completeness for the restaurant data set and Figure 6.4 shows the same for the census data set.

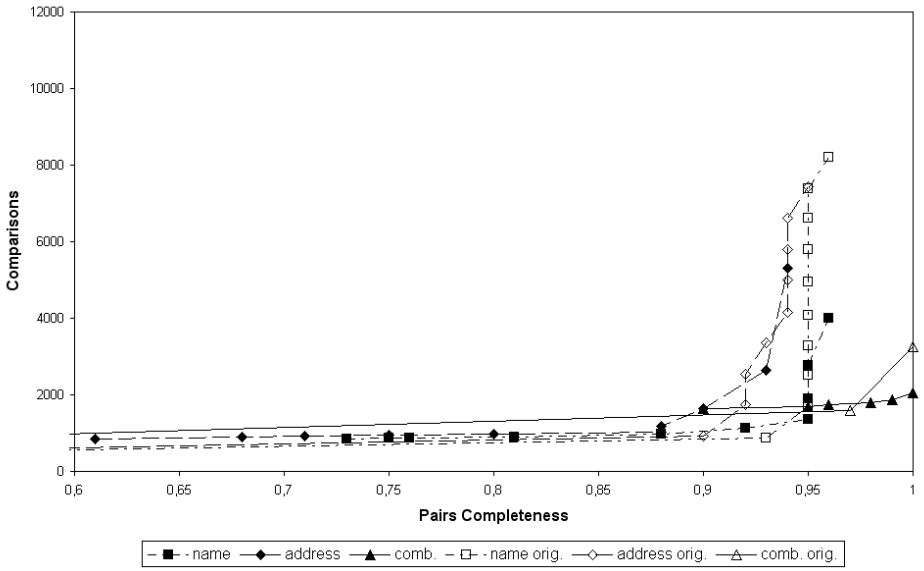

Figure 6.3: Blocking Costs for the Restaurant Data Set

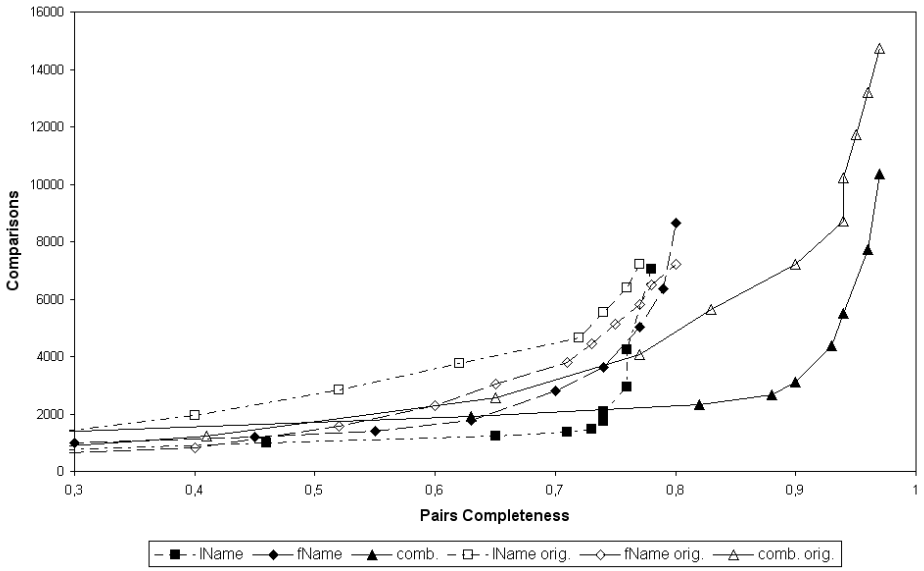

Figure 6.4: Blocking Costs for the Census Data Set

These figures show that the proposed approach with dynamically sized windows is either equally expensive or even cheaper in terms of number of comparisons. Again the most interesting point is the point of maximum pairs completeness. There the combined approaches on the restaurant data set show 2048 comparisons for this vs. 3244 comparisons for the original approach; on the census data set the approach needs 10365 vs. 14732 comparisons for the original approach for the maximum pairs completeness. This reveals that moving a part of the comparison complexity already to the blocking phase does not only significantly increase the accuracy of the resulting candidate duplicates, but also reduces the overall costs of the duplicate detection process.

6.2.3 Comparison with Other Blocking Methods

Baxter et al. [5] have compared the standard blocking using keys, the original sorted-neighborhood, the bigram indexing and the canopy clustering methods against the mailing data set. In order to compare the dynamic sorted-neighborhood approach with their results, it is conducted on the same data set with three different dynamic window sizes: 0.1, 0.2 and 0.3.

Figure 6.5 shows the pairs completeness, Figure 6.6 shows the reduction ratio and Figure 6.7 shows the F score against the size of the data sets for the three window sizes and the two best other approaches from [5], that are the canopy clustering with cluster size 1.5 and bigram indexing with bigram size of 0.3.

These figures clearly show that the proposed approach is competitive with the best other approaches. The different window sizes show no significant difference here and can therefore be ignored. For all data sets the dynamic sorted-neighborhood method is the best approach in terms of reduction ratio. For smaller data sets (1000 and 2000 records) the approach even outperforms the others in terms of pairs completeness, whereas for larger data sets the pairs completeness gets worse compared to canopy clustering. In contrast to the other top blocking methods, the approach does not require any new indexing structures to be implemented. It simply uses existing standard indices on individual attributes.

The bigram indexing behaves unstable for the smaller data sets (1000 and 2000 records), but becomes more predictable for the larger data set. A reason for this behavior is not given in the literature [5].

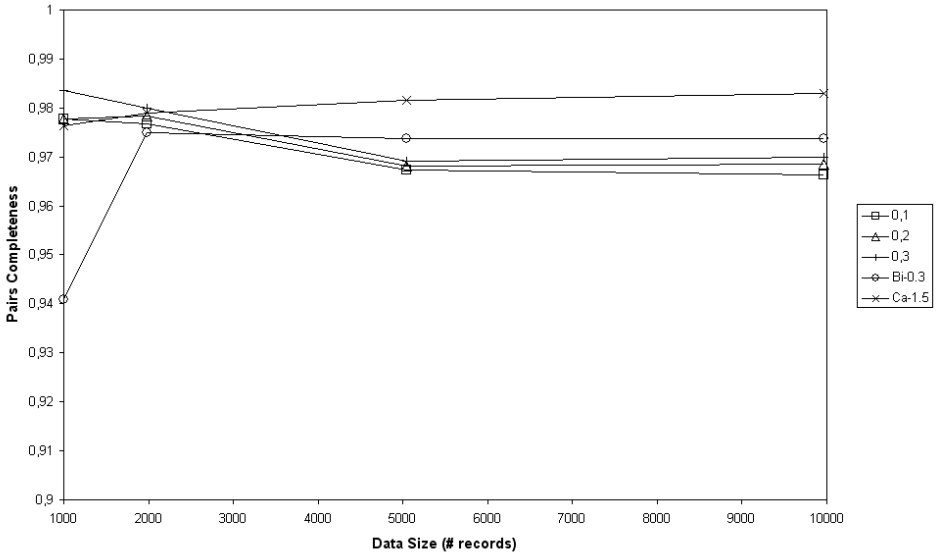

Figure 6.5: Pairs Completeness on the Mailing Data Set

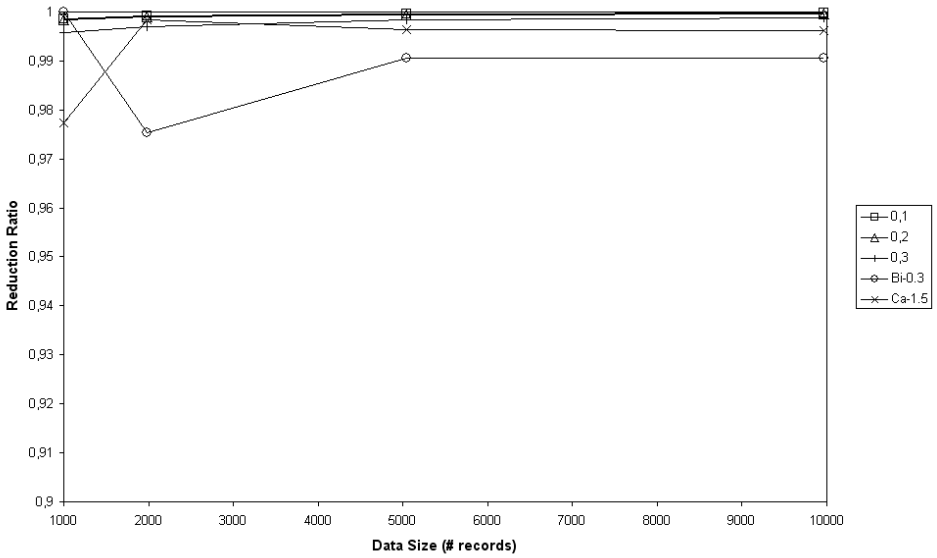

Figure 6.6: Reduction Ratio on the Mailing Data Set

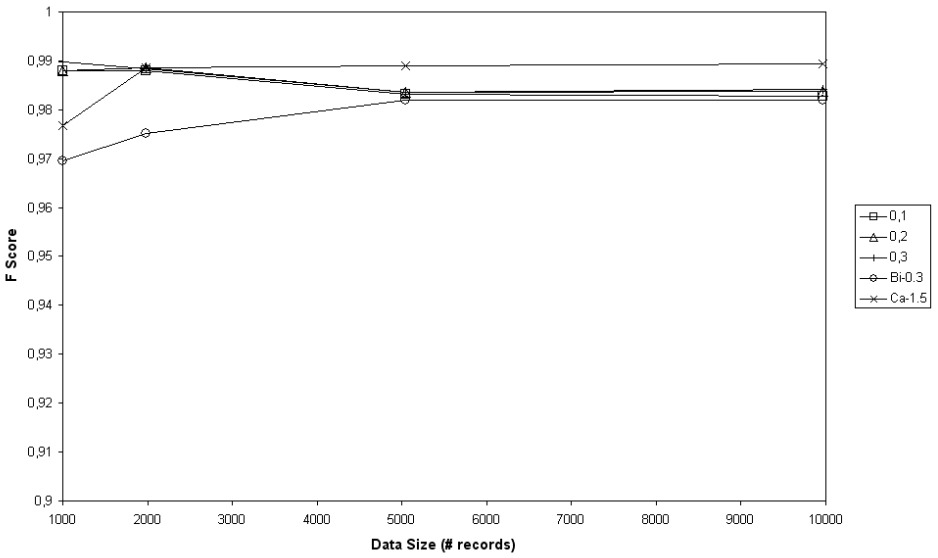

Figure 6.7: F Score on the Mailing Data Set

6.2.4 Scalability

For evaluating performance and scalability the blocking algorithm is also tested on the *Publication* data sets. These contain overall 969364 publications with 879154 distinct titles. Additionally to the full data set, also versions containing only 50%, 25% and 10% of the publications are used for comparison.

The blocking is done with three different window sizes (0.15, 0.2, 0.25) on the title attribute. The experiments were conducted on a Pentium 4/3.2GHz dual core machine with 2GB RAM. The algorithm simultaneously finds candidate duplicates between the data sets and the non-duplicates within the data sets. Table 6.5 shows the number of found candidate duplicates, non-duplicates, performed title comparisons and the overall process time depending on the window size. Figure 6.8 shows the relationship between the number of publications to the actual processing time and Figure 6.9 the relationship between the number of candidate duplicates that were actually found for the differently sized data sets to the actual processing time.

These experiments clearly show that the algorithm is also scalable for large real world data sets. About 10 minutes for the full data set is absolutely acceptable, because duplicate detection on such a large data set is typically

Table 6.5: Performance of the Blocking algorithm

Window	Cand. Duplicates	Non-Duplicates	Comparisons	Time
100% (969364 publications, 879154 titles):				
0.15	136684	1420301	965475	496 sec
0.20	194492	1524653	1086687	637 sec
0.25	510714	2051674	1751239	978 sec
50% (483960 publications, 458837 titles):				
0.15	34731	347244	484364	210 sec
0.20	54180	381682	530112	236 sec
0.25	166395	565800	790182	362 sec
25% (242269 publications, 235199 titles):				
0.15	8962	81276	242340	98 sec
0.20	15363	92580	258415	106 sec
0.25	56303	158155	357989	147 sec
10% (96691 publications, 95351 titles):				
0.15	1440	12691	96703	37 sec
0.20	2927	15193	100561	40 sec
0.25	13861	32524	127996	52 sec

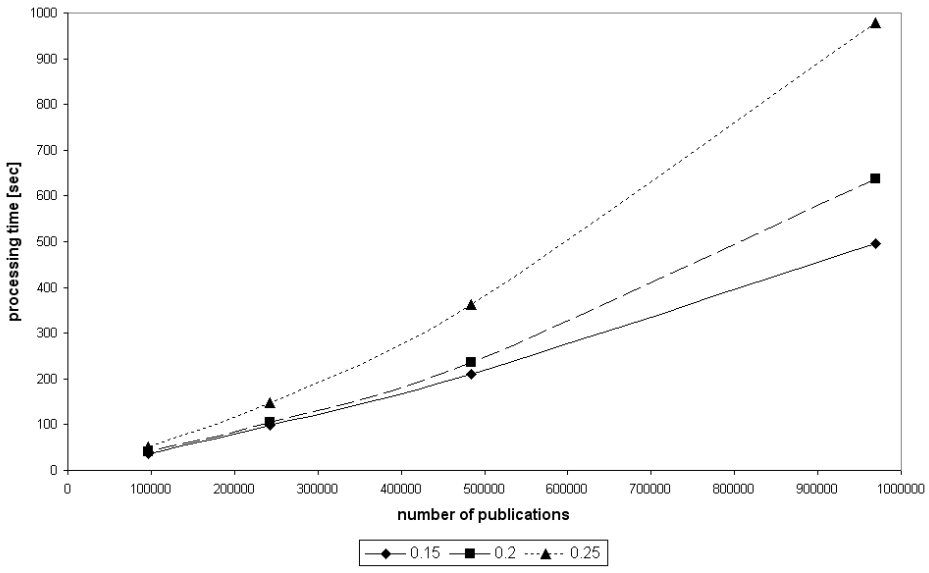

Figure 6.8: Scalability of the Blocking Algorithm in respect to the number of publications

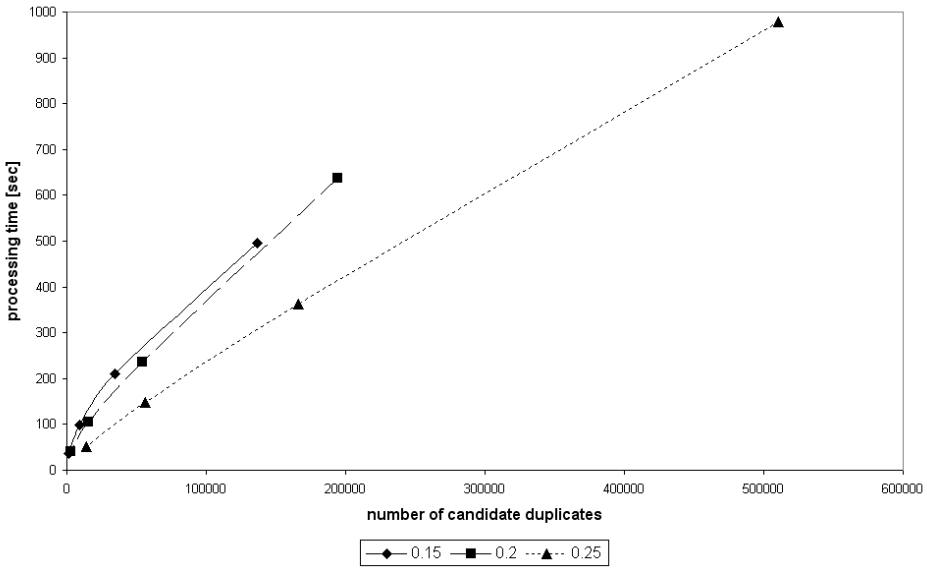

Figure 6.9: Scalability of the Blocking Algorithm in respect to the number of candidate duplicates

done offline.

The evolution of the processing time from the smaller to the larger data sets shows basically a linear behavior that is based on the linear increasing number of key comparisons. The number of detected candidate duplicates and non-duplicates increases quadratically with the size of the data sets, which has also a small quadratic influence on the processing time. The duplicate searching itself shows even a slightly sublinear behavior as can be seen in Figure 6.9.

6.3 Matching Experiments

This section presents the evaluation results of the matching approach. After introducing the relevant settings and measures in the following section, the developed extensions for the Fellegi-Sunter model are evaluated by comparing them to the original methods in Section 6.3.2. In Section 6.3.3 the machine learning methods are used as decision model and then Section 6.3.4 compares these results with the Fellegi-Sunter approach, a simple baseline method and related unsupervised and supervised approaches. Finally, Section 6.3.6 analyzes the impact of different similarity measures used in the

comparison module on the overall matching accuracy.

6.3.1 Experimental Methodology

For the blocking phase the multi-pass algorithm as described in Section 3 is used with a window distance size of 0.25. On the restaurant data set the blocking was done on the name and address attribute resulting in 251 candidate duplicate pairs for the set M', which corresponds to 100% recall and 45% precision, and 188 non-duplicate pairs for the set U'.

On the census data set the blocking was done on the last name and first name attribute resulting in 1524 candidate duplicate pairs for the set M', which corresponds to 90% recall and 19% precision, and 1607 non-duplicate pairs for the set U'. 100% recall was not reached for the census data set, because the remaining duplicates were too different in the beginning character of the last and first name, therefore these duplicates did not fall into the same window, e.g., "Hillion, George" and "Killion, Jorge". These duplicates would also be difficult to identify for humans.

On the publication data set the blocking was done on the title attribute with a window distance size of 0.15 resulting in 21142 candidate duplicate pairs for the set M', which corresponds to 100% recall and 95% precision. Further 79811 non-duplicate pairs are generated for the set U'. For the counting in the Fellegi-Sunter model as well as for the training in the SVM model only 1000 random sample pairs from M' and U' are used.

For comparison of the experimental results precision, recall and F-measures. are calculated. These are defined as usual in information retrieval [3]:

$$Precision = \frac{|CorrectlyIdentifiedDuplicates|}{|IdentifiedDuplicates|}$$

$$Recall = \frac{|CorrectlyIdentifiedDuplicates|}{|TrueDuplicates|}$$

$$F - measure = \frac{2 * Precision * Recall}{Precision + Recall}$$

The precision-recall curves in the figures use interpolated precision values at 20 standard recall levels following the traditional procedure in information retrieval [3]. However, the figures show only the interesting area between the recall levels 0.6 and 1.

Table 6.6: Maximum F-measures using the Fellegi-Sunter model as decision model

Method	Restaurant	Census	Publication
Baseline	0.935	0.725	0.976
F&S independent	0.939	0.840	0.993
F&S dependent	0.938	0.883	0.989

6.3.2 The Fellegi-Sunter Model as Decision Model

The extensions for the Fellegi-Sunter model are evaluated separately for the effect of taking dependencies into account, using continuous comparison values and the impact of the parameter $P(M)$.

Dependencies A first experiment compares the precision and recall of the Fellegi-Sunter model with and without the independence assumption. This is done using continuous values as described in Chapter 4. Furthermore, the results are also compared against a simple baseline, which simply concatenates the string values of the individual attributes and takes the string similarity between these concatenated strings as overall result - ignoring attribute relevance.

The maximum F-measures of these methods on both data sets are shown in Table 6.6. Figure 6.10 shows the precision and recall curves for the restaurant data set, Figure 6.11 the curves for the census data set and Figure 6.12 the curves for the publication data set.

These results show that the Fellegi-Sunter model taking dependencies into account always reaches one of the highest accuracies. However, the effect of taking dependencies into account depends on the data set, i.e., if there exists stronger dependencies between the attribute similarities, the accuracy taking dependencies into account is significantly higher than using the independence assumption. This can be clearly seen for the census data set. In addition, even under the independence assumption the results are significantly better than the baseline experiment.

Continuous Values The previous experiment has already used the extension for continuous distances measures. In order to show the impact of these continuous values in contrast to thresholded boolean "match" - "not match" values is shown in the following experiments. The results of the Fellegi-Sunter model with continuous values and taking dependency into account is compared to the results using thresholded boolean values, with

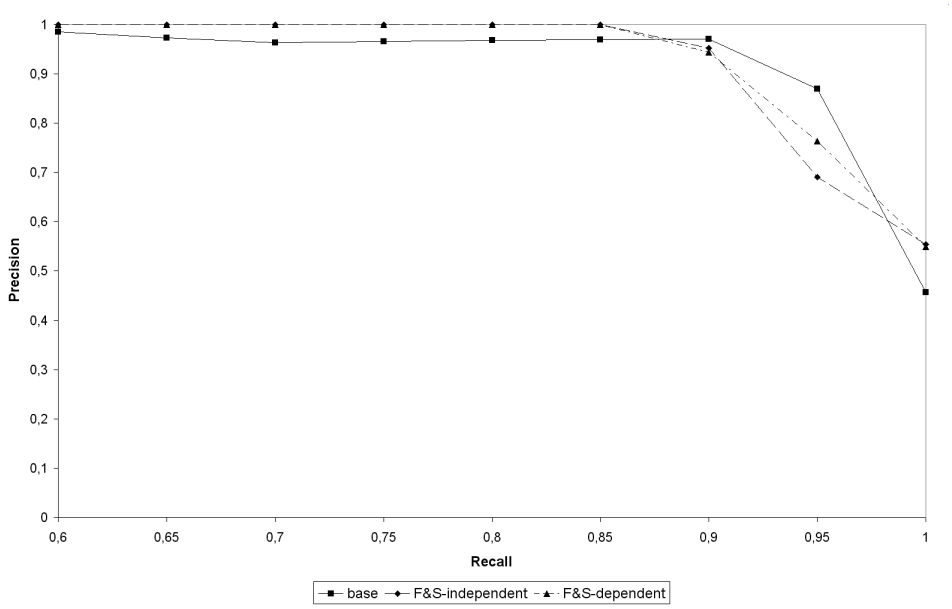

Figure 6.10: Precision-Recall for the restaurant data set

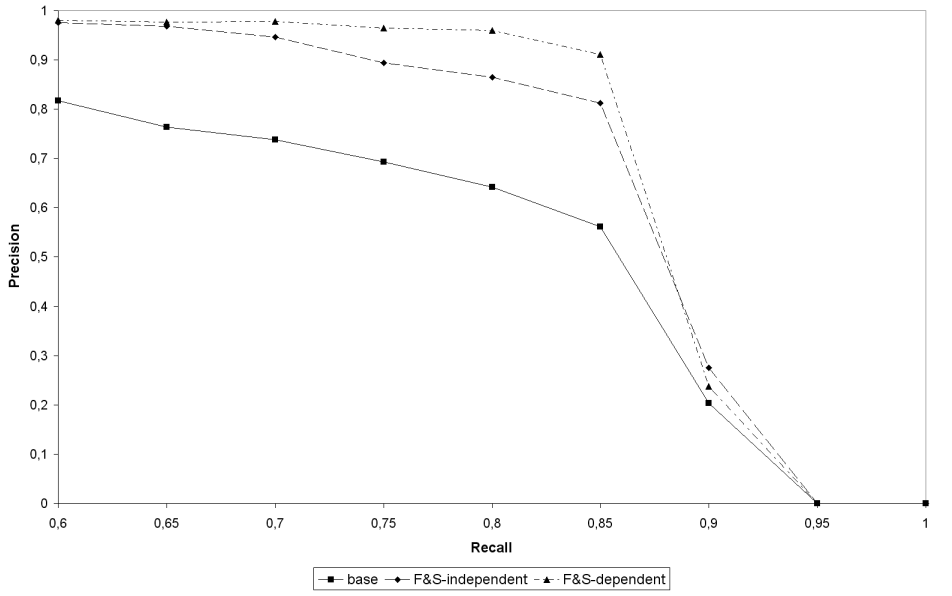

Figure 6.11: Precision-Recall for the census data set

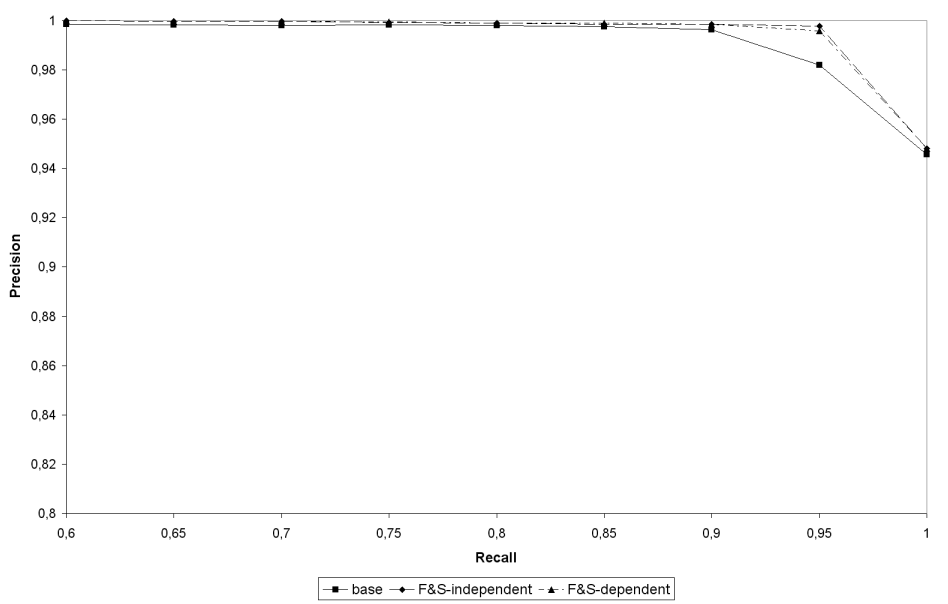

Figure 6.12: Precision-Recall for the publication data set

Table 6.7: Maximum F-measures for boolean variables

Method	Restaurant	Census	Publication
F&S continuous	0.938	0.883	0.989
boolean - 0.1	0.845	0.503	0.984
boolean - 0.2	0.895	0.789	0.992
boolean - 0.3	0.914	0.875	0.992
boolean - 0.4	0.902	0.803	0.986

thresholds of 0.1, 0.2, 0.3 and 0.4. The maximum F-measures of these methods on both data sets are shown in Table 6.7. Figure 6.13 shows the precision and recall curves for the restaurant data set, Figure 6.14 the curves for the census data set and Figure 6.15 the curves for the publication data set.

These results show that boolean values can reach nearly as good results as continuous values under the condition that they are well calibrated. Poorly calibrated thresholds for the boolean values can result in very poor results, like the results for the thresholds 0.1 or 0.4. Smaller thresholds generally result in better precision, but lower recall. Therefore the use of continuous distance measures for the comparison vector is not only highly accurate, but also requires much less user-interaction than using boolean

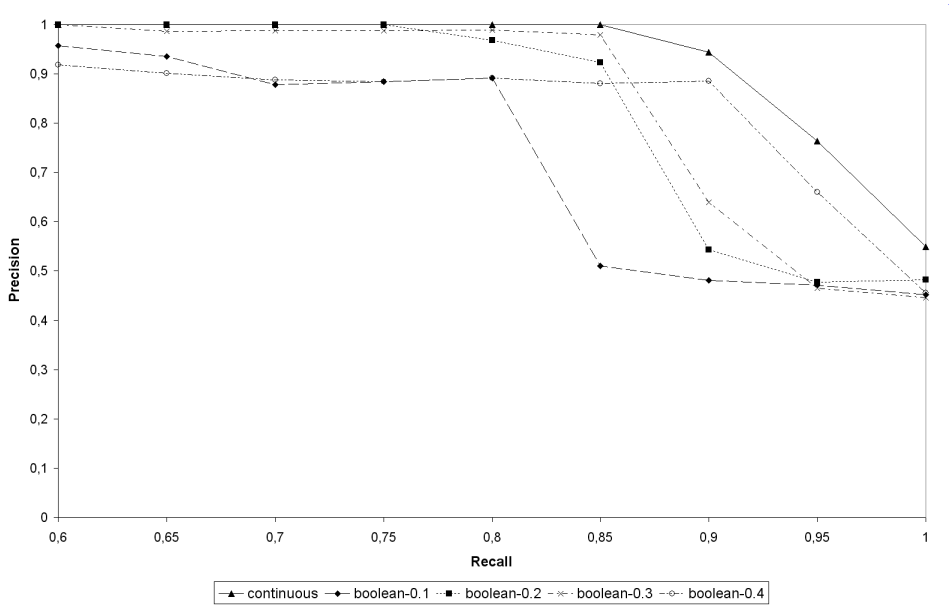

Figure 6.13: Precision-Recall for boolean variables on the restaurant data set

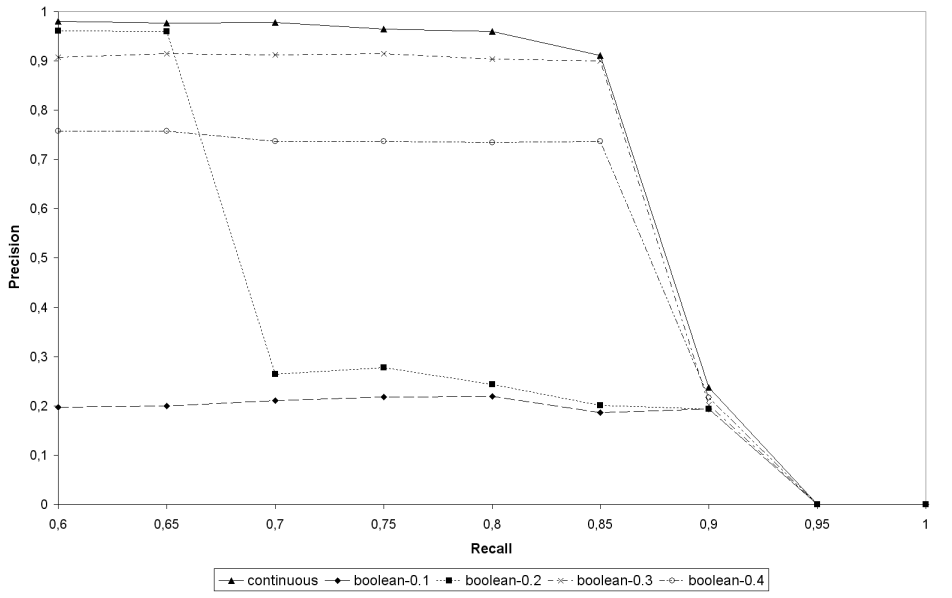

Figure 6.14: Precision-Recall for boolean variables on the census data set

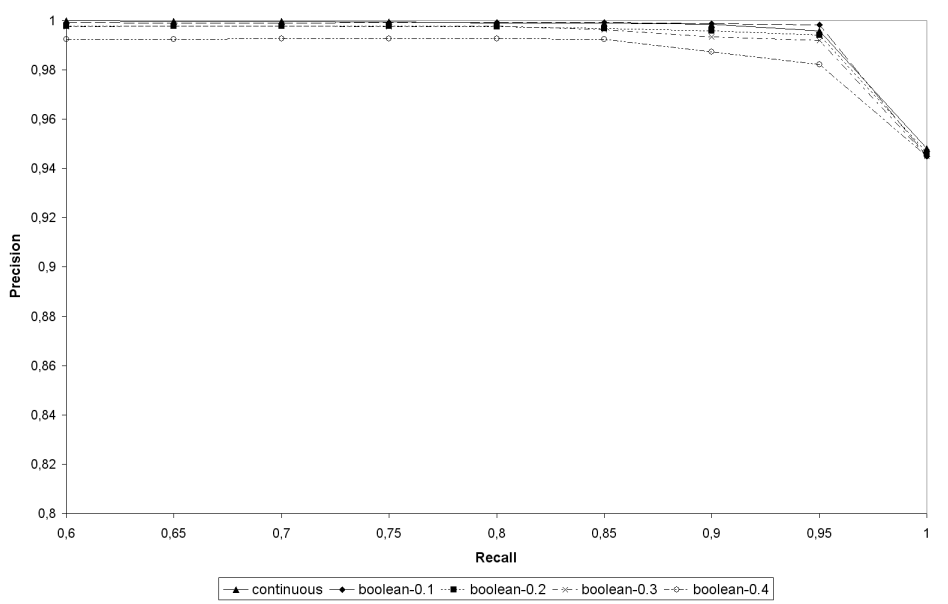

Figure 6.15: Precision-Recall for boolean variables on the publication data set

values and is therefore preferable.

Impact of the Parameter $P(M)$ The parameter $P(M)$ as defined in Section 4.1 defines the general expected probability for duplicates within the examined data set. This parameter scales the final duplicate estimation value, i.e., if only few non-duplicates are expected, the result will also be scaled higher. This should enable applications to set a threshold on the final result independent of the data.

In order to evaluate the impact of this parameter, three different methods for determining $P(M)$ are compared:

- default: $P(M)$ is set to 0.5, i.e., it is equally probable to find a duplicate and a non-duplicate in the set of candidate duplicates. This method is equal to the original Fellegi-Sunter ratio without the parameter $P(M)$ and using a threshold of 1.
- estimated: $P(M)$ is iteratively estimated as described in Section 5.3.1.
- true: $P(M)$ is set to the actual ratio of the number of duplicates within the candidate duplicate set to the number of all candidates.

Table 6.8: Impact of the Parameter $P(M)$ on the F-measure

Method	Restaurant	Census	Publication
default	0.821 (0.5)	0.555 (0.5)	0.914 (0.5)
estimated	0.767 (0.697)	0.570 (0.458)	0.988 (0.956)
true	0.911 (0.446)	0.879 (0.194)	0.988 (0.945)

The f-measures for this experiment determined at the natural threshold (50%) are shown in Table 6.8. The actually used value for $P(M)$ on the particular data set is given in parentheses.

This experiment shows that if the true $P(M)$ value significantly differs from 0.5 the impact of $P(M)$ is high. In this cases also the proposed method to determine this value increases the f-measure. However, in cases, where the true $P(M)$ is near 0.5 it may even worsen the result, e.g., for the restaurant data set. It can also be seen that the estimated value for $P(M)$ is always an overestimation, this is caused by using the full M' for determining $m(\gamma)$, which is therefore also an overestimation. This overestimation of $P(M)$ is relatively higher the more non-duplicates are present in M'. This suggests to estimate the $P(M)$ value using a cleaned version of M' for determining $m(\gamma)$.

6.3.3 Machine Learning Methods as Decision Model

KMeans A first experiment uses the KMeans method on the data sets initializing the cluster centroids for the duplicate cluster with the mean of M' and for the non-duplicate cluster with the mean of U'. The used KMeans algorithm gives no confidence into its decision if an object belongs to the duplicates or the non-duplicates. Therefore no precision-recall curves can be drawn for this experiment, but only the precision and recall for the duplicate cluster can be used as evaluation criterion.

This configuration performs quite poor, the f-measure for the restaurant data set is 0.830 and for the census data set only 0.01. However, it is interesting to note that when holding the centroid for the non-duplicates constant, the f-measure increases on the restaurant data set to 0.890, but this has no significant effect on the census data set.

This can be explained with the distribution of the clusters is neither spherically shaped nor linearly separable and therefore KMeans is not well suited for this scenario.

SVM The second experiment uses the support vector machines in the way as described in Section 5.3.2, i.e., the SVM is trained with U' as negative examples and with M' as positive examples. To this end the *libsvm* library for support vector machines was used as provided by [14], taking c-svcs, with a radial basis kernel and setting the gamma and cost parameter to 1. This results in maximum f-measures for the restaurant data set of 0.964, for the census data set of 0.587 and for the publication data set of 0.993.

This shows that support vector machines can provide a very high accuracy, given that the number of non-duplicates in the set M' is not too high, which is the reason, why it failed on the census data set. However, when using the Fellegi-Sunter model beforehand to filter M', then this results in maximum f-measures for the restaurant data set of 0.960, for the census data set of 0.907 and for the publication data set of 0.993. These results are very convincing and show the highest accuracy of all unsupervised approaches. The next section compares the results with the other approaches and presents the precision-recall curves.

It is interesting to note, that these results were achieved using default values for the gamma and cost parameters of the SVM. This is a quite unusual way of using SVMs as typically the results depend on these settings and need to be calibrated based on the training sets. However, experiments that tried to calibrate these parameters using M' and U' result in significantly worse accuracy.

These experiments were conducted using a usual 10-fold cross validation. This divides the training sets M' and U' into 10 equally sized sets that can be further combined to 10 different test and training sets, i.e., always one part is used as test set and the union of the other 9 sets are used as training sets. Then several combinations of values for the gamma and cost parameter are tested on all of these 10 test and training set combinations, calculating the average number of false matches and false misses for each parameter setting. The parameter setting that results in the lowest number of average false matches and false misses are then used for the actual training.

The reason for the worse accuracy of these experiments using calibrated parameters seems to be the dirtiness of the M' training set.

6.3.4 Comparison of the Approaches

This section compares the Fellgi-Sunter based and SVM based approaches with other state of the art methods. The following approaches are compared:

- Base: this baseline simply concatenates the individual string values of

Table 6.9: Maximum F-measures for detecting duplicates

Method	Restaurant	Census	Publication
Base	0.935	0.725	0.976
Fellegi-Sunter	0.938	0.883	0.989
SVM (+Fellegi-Sunter)	0.960	0.907	0.993
HGM (SoftTFIDF)	0.844	0.759	-
SVM B&M (Jaccard)	0.971	-	-

the attributes and takes the string similarity between these concatenated strings as overall result - ignoring attribute relevance.

- HGM (Hierarchical Graphical Model): this is the unsupervised approach presented in [65] that uses the same data sets for their evaluation. Their results are simply copied for comparison, although they used a different blocking algorithm. (only for the restaurant and census data set)

- Fellegi-Sunter: The Fellegi-Sunter based approach using continuous distance values and taking dependencies into account.

- SVM (+Fellegi-Sunter): The SVM based approach, using the Fellegi-Sunter model beforehand to filter the initial set M'.

- SVM B&M (Jaccard): this is the best of the fully supervised approaches presented by Bilenko & Mooney in [10] that uses the same restaurant data set for their evaluation. Their results are directly copied from the publication.

It must be noted that the Jaro distance is used as distance function for the experiments as provided by [22], whereas [65] was using the SoftTFIDF distance and [10] was using the Jaccard distance. A comparison of these distance functions is presented in [23]. The impact of different distance functions is evaluated in Section 6.3.6.

The maximum F-measures of these methods on both data sets are shown in Table 6.9. Figure 6.16 shows the precision and recall curves for the restaurant data set, Figure 6.17 the curves for the census data set and Figure 6.18 shows the curves for the publication data set.

These results clearly show that the proposed unsupervised methods significantly outperform other existing unsupervised methods and they are even not far away from the results of fully supervised methods. In particular,

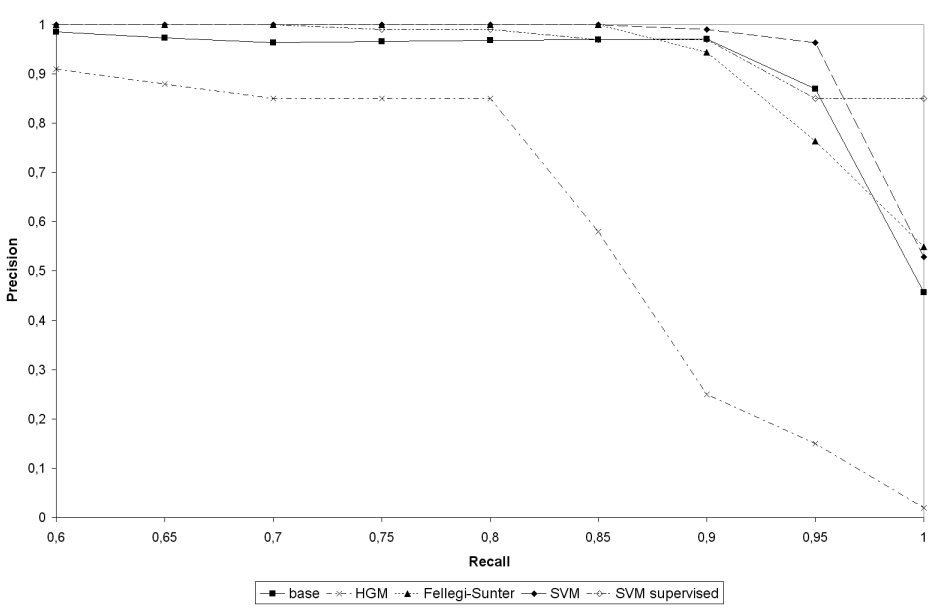

Figure 6.16: Precision-Recall for the restaurant data set

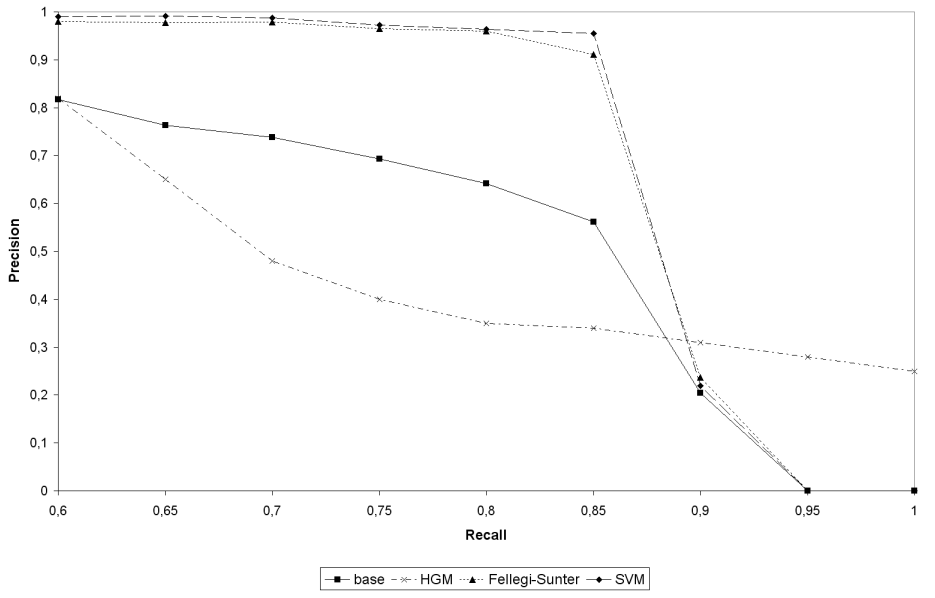

Figure 6.17: Precision-Recall for the census data set

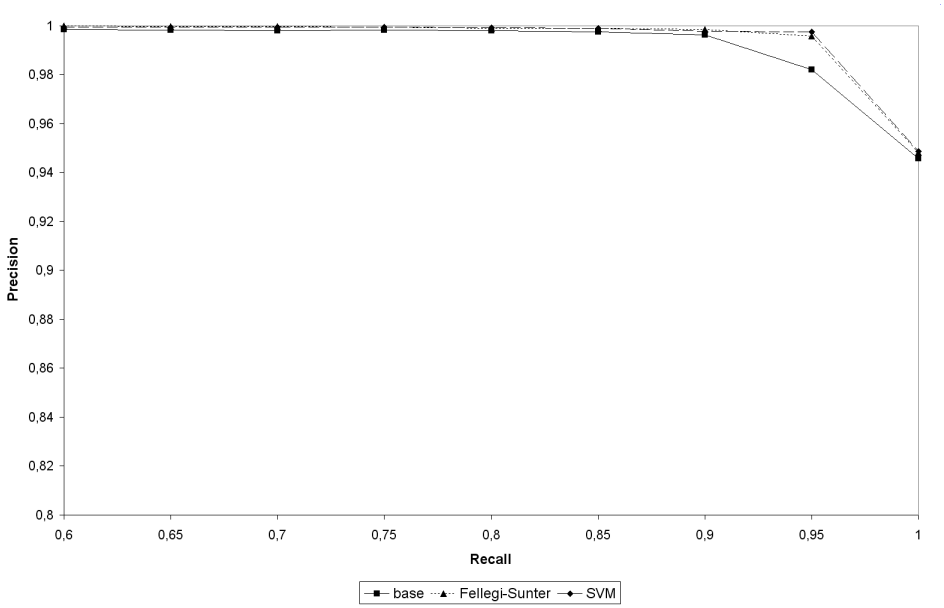

Figure 6.18: Precision-Recall for the publication data set

the supervised SVM method can only outperform the unsupervised SVM method of this thesis at very high recall levels (above 0.95). The unsupervised HGM method performs always significantly worse, interestingly, it shows even a lower accuracy than the simple baseline approach. The SVM based approach shows always slightly better accuracy than the Fellegi-Sunter approach, which shows that SVMs can model the distributions of the comparison vectors better than even the extensions for the Fellegi-Sunter model. It can be further seen that in particular for quite dirty data sets like the census, the simple baseline method performs very poor in contrast to the proposed methods.

The higher precision of the HGM method for the census data set at the 100% recall level is caused by a different blocking algorithm that reaches here a higher recall to the disadvantage of a very poor precision. Ours reaches here around 90% recall, therefore the precision for the 95% and 100% recall levels are 0.

6.3.5 Scalability

In order to evaluate the scalability of the decision models the labeled subset of the publication data set is used. Additionally to the full data set, also

Table 6.10: Performance of the Decision Models

Decision Model	Max. F-Measure	Time
100% (62403+56340 publications, 21142 candidate duplicates):		
F&S	0.989	60.7 sec
SVM	0.993	13.4 sec
50% (31108+28112 publications, 5347 candidate duplicates):		
F&S	0.985	15.8 sec
SVM	0.990	3.3 sec
25% (15638+14044 publications, 1310 candidate duplicates):		
F&S	0.988	4.6 sec
SVM	0.990	2.1 sec
10% (6267+5492 publications, 210 candidate duplicates):		
F&S	0.987	0.9 sec
SVM	0.990	0.5 sec

versions containing only 50%, 25% and 10% of the publications are used for comparison. The matching is done with the Fellegi-Sunter model taking dependencies into account and using continuous comparison values and the SVM based approach. The counting and training is done on 1000 sample records of the candidate duplicates and 1000 sample records of the non-duplicates.

The experiments were conducted on a Pentium 4/3.2GHz dual core machine with 2GB RAM. Table 6.10 shows the number of input candidate duplicates, the resulting maximum f-measure and the overall process time depending on the decision model. Figure 6.19 shows the relationship between the number of candidate duplicates to the actual processing time.

The experiments show that both algorithm scale linearly with the number of candidate duplicates. As the number of candidate duplicates (and true duplicates) increase quadratically with the number publications, the overall complexity is quadratic.

6.3.6 Impact of Similarity Measures

The previous experiments used the Jaro distance for comparison. In order to assess the impact of different similarity functions, the experiments using the Fellegi-Sunter model and the SVM are also carried out with a simple Levenshtein distance [50] and a TFIDF (term frequency/inverse document frequency) distance as often used in information retrieval. Additionally on the restaurant data set a combination of the Jaro and the TFIDF similarity is

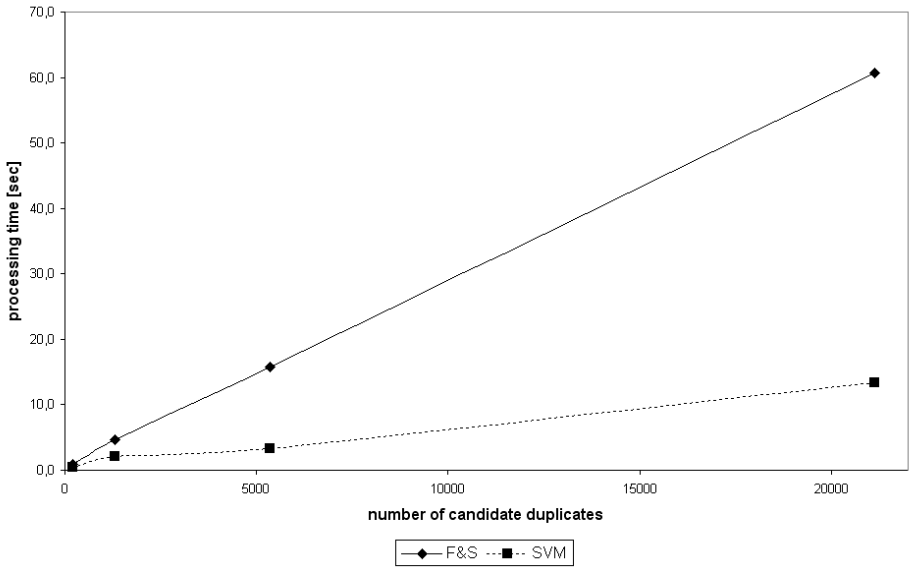

Figure 6.19: Scalability of the Decision Models

tested. The maximum F-measures can be found in Table 6.11 the precision-recall curves can be found in Figure 6.20 and Figure 6.21 for the restaurant data set and in Figure 6.22 and Figure 6.23 for the census data set.

The experiment shows that the selection of a good similarity function is of great importance. It shows that the effectiveness of a similarity function depends on the data, in particular the TFIDF similarity function performs much better on the restaurant data set than on the census data. The results further show that the SVM approach is much more sensible to a good similarity function, both alternative similarity functions result in a much worse accuracy. The combination of several similarity measures may outperform those with just one, if both similarity measures detect some and different kind of errors, like in the restaurant data set. However, simply adding several similarity measures to a comparison vector independent if this particular similarity measure suits the possible errors in this data, decreases the overall accuracy as can be seen for the census data set. These results demand for further work in unsupervised finding ideal similarity functions.

Table 6.11: Maximum F-measures using different distance measures

Method	Restaurant	Census
F&S levenshtein	0.909	0.879
F&S jaro	0.938	0.883
F&S tfidf	0.959	0.625
F&S tfidf+jaro	0.963	0.795
SVM levenshtein	0.949	0.563
SVM jaro	0.960	0.907
SVM tfidf	0.978	0.699
SVM tfidf+jaro	0.969	0.702

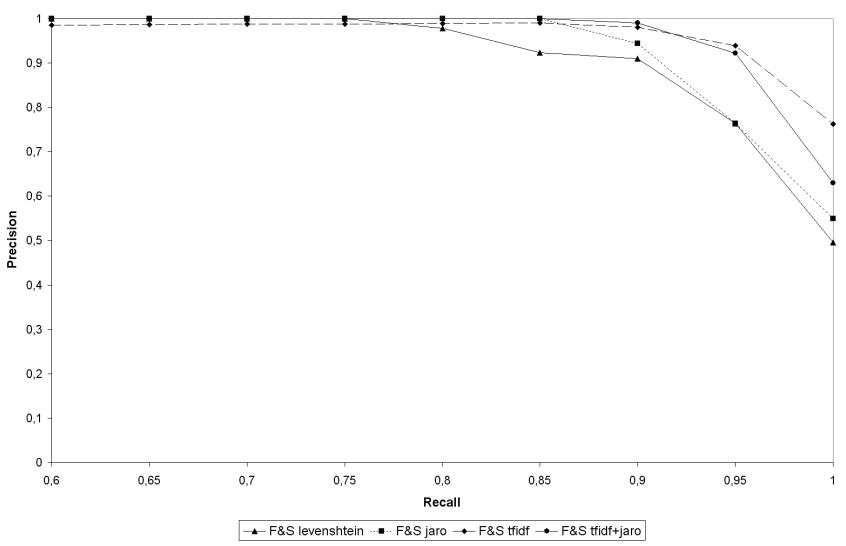

Figure 6.20: Precision-Recall for the restaurant data set using the Fellegi&Sunter model with different similarity measures

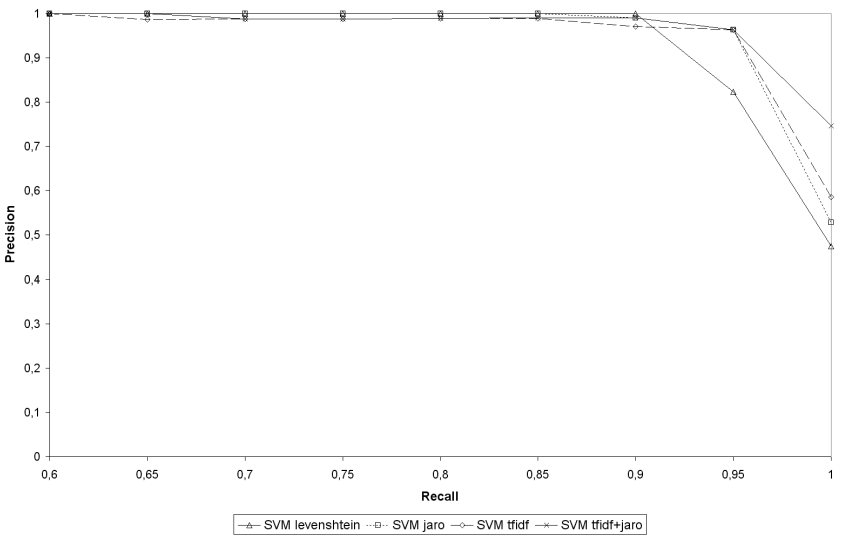

Figure 6.21: Precision-Recall for the restaurant data set using the SVM model with different similarity measures

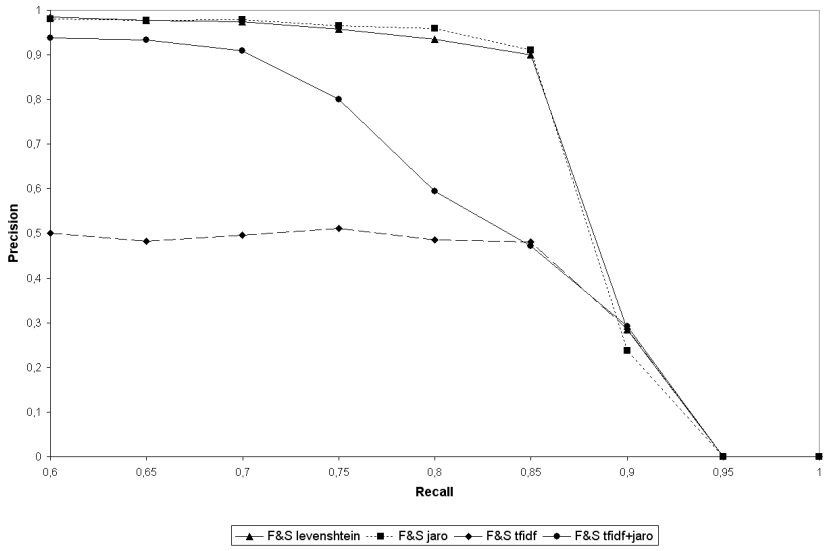

Figure 6.22: Precision-Recall for the census data set using the Fellegi&Sunter model with different similarity measures

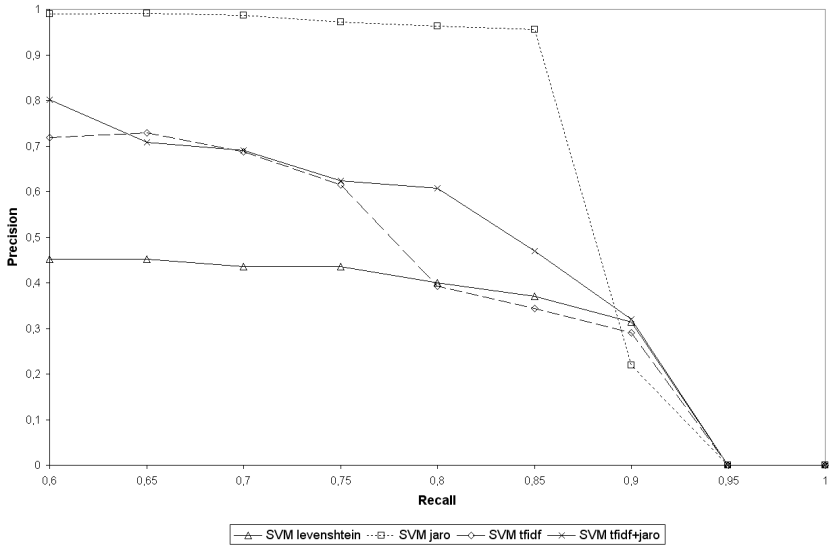

Figure 6.23: Precision-Recall for the census data set using the SVM model with different similarity measures

6.4 Summary

This chapter has presented the results of the experiments of both the blocking method and the matching method. The results can be summarized as follows:

1. Blocking
    - The developed blocking algorithm produces less false matches and less false misses than the original sorted-neighborhood method.
    - It is also less or equally costly compared to the original sorted-neighborhood method.
    - Identical to the original method, the use of multiple passes with different keys significantly increases the accuracy of the resulting candidate duplicates.
    - The approach is competitive to other state of the art blocking methods.
    - It is also scalable to large real world data sets.
2. Matching

- The presented extensions for the Fellegi-Sunter model for continuous distance values and taking dependencies into account outperform the classic approach.
- The introduced parameter $P(M)$ for the Fellegi-Sunter model helps to find a domain independent threshold on the final duplicate estimation value.
- The SVM and Fellegi-Sunter based decision models outperform other state of the art unsupervised models and are even almost as good as fully supervised methods.
- The SVM method is always slightly more accurate than the extended Fellegi-Sunter model.
- The SVM and Fellegi-Sunter based decision models are also scalable to large data sets.
- The applied similarity measure has an enormous impact on the results of the decision module.

These results suggest the following settings for duplicate detection. The blocking method should be used with multiple passes on all attributes that are sufficiently unique. The distance measure depends on the used sorting method, in these experiments the Jaro distance on an alphabetically sorted index has been used. For this distance measure window sizes between 0.15 and 0.25 have shown to be a good choice.

For the matching phase the SVM based approach has shown the highest accuracy and good scalability, however, this method is not as robust as the Fellegi-Sunter based approach, when the candidate duplicates contain many non-duplicates or if poor performing similarity measures are chosen. Therefore, it is suggested to use the Fellegi-Sunter model in advance to filter the initial training set M' for the SVM. When using the Fellegi-Sunter model, the proposed extension for taking dependencies into account and using continuous comparison values perform in general better than the classic methods. However, if the independence assumption is approximately valid between the attributes, the Fellegi-Sunter model using this assumption can perform equally well and significantly simplifies the approach and implementation.

The used similarity measures for the individual attributes should be selected carefully, because of their enormous impact on the final results of the decision module. In the experiments the Jaro measure has shown to be a good default choice. In particular, the Fellegi-Sunter model is able to profit from several similarity measures for the same attribute, whereas the

SVM approach is sensible to differently scaled similarity values within the comparison vector.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Contribution This thesis has presented ways to detect duplicates in the context of integrating two individual data sources with only little human interaction. To this end a precise blocking method is used to automatically extract a set of candidate duplicates and a set of sample non-duplicates from the two data sources. These sample non-duplicates are then used to filter out the remaining non-duplicates from the set of candidate duplicates, resulting ideally in the clean set of duplicates.

The presented blocking method is an extension of the classic sorted-neighborhood method, using a dynamically sized window based on a distance measure in contrast to the fixed window in the original approach. Additionally, it iterates over the list of distinct keys instead of the list of records itself, which results in less key comparisons. This blocking method has shown to be more accurate and less costly than the original sorted-neighborhood method. Furthermore, it is competitive with other state of the art methods, like the canopy clustering and the bigram indexing, but in contrast to these methods it needs only standard indices instead of special purpose indices to work efficiently. Experiments have also shown that the algorithms complexity is linear with the size of the data set, which allows it to scale also to very large data sources.

This blocking algorithm is used typically in multiple passes with different keys, which increases the accuracy of the resulting candidate duplicates. Additionally to the set of candidate duplicates, the same blocking algorithm is also used on the individual data sources, to extract non-duplicates, which still show some and the same kind of similarity as the non-duplicates that

are present in the candidate duplicates. This is based on the assumption that the individual data sources are more or less duplicate free.

In a second step all candidate duplicates and non-duplicates are compared in detail by a comparison module. This uses similarity measures for every attribute-pair and results in a vector of such similarity values, which is called the comparison vector. These sets of comparison vectors form the input to the following decision model, which should now be able to classify the comparison vectors of the candidate duplicates into actual duplicates and non-duplicates.

Several decision models could be used for this task. This thesis has used the classic Fellegi-Sunter model from the statistics community and has extended it to handle also continuously valued similarity measures and taking dependencies into account. Experiments have shown that these extensions in general provide higher accuracy than the classic approach, only for data sources where the independence assumption more or less holds, the extension for taking dependencies shows no significant effect, but only causes higher complexity of the algorithm. Using the extended Fellegi-Sunter model with the provided set of candidate duplicates and non-duplicates results in a duplicate detection accuracy that clearly outperforms other unsupervised approaches and is also not far from supervised approaches.

Besides of the Fellegi-Sunter model as decision model, also methods from the machine learning community are used. So far, Support Vector Machines (SVM), a state of the art classification method that is typically used in supervised settings are used as decision model, using the candidate duplicates and non-duplicates as initial training sets. The set of candidate duplicates is iteratively cleaned from non-duplicates by removing pairs classified as non-duplicates in the previous iteration step. This approach provides even better accuracy than the Fellegi-Sunter based approach, at least as long as the set of candidate duplicates contains not too many non-duplicates. In this case the SVM based approach might fail completely, i.e., it results in very poor accuracy. This case can be handled by using the Fellegi-Sunter model beforehand to filter the initial set of candidate duplicates. This combined approach is always more accurate than the Fellegi-Sunter model and comes very close to fully supervised methods.

In addition to classification methods like SVMs, it is also possible to see the decision task as clustering problem. To this end, the classic KMeans clustering approach was also used for this. However, this algorithm assumes spherical shaped clusters, which is typically not the case for the comparison vectors of duplicates and non-duplicates that often show even dependencies between the individual comparison vector components, i.e., dependencies

between the attributes. This seems to be reason, why this algorithm shows a very poor accuracy in the conducted experiments.

Based on these experiments it is possible to conclude that in general the SVM based approach is the best choice, possibly combined with the Fellegi-Sunter based approach for filtering the candidate duplicates beforehand if it is expected that these contain many non-duplicates. If different similarity measures are used within the comparison vector it is important to equally scale these measures, because the SVM behaves sensible to differently scaled similarity measures.

Further experiments have analyzed the impact of the used similarity measures during the comparison phase on the overall duplicate detection accuracy. The results show that this impact is enormous and therefore these similarity measure must be chosen carefully. The Jaro distance has shown good universality properties, but others might provide significantly better results depending on the data and the typical errors in the data. In some cases also a combination of several similarity measures for the same attribute-pair provide the best results.

If not only flat records are used as data set, but also relationships between objects occur, an additional family of similarity measures can be used. These similarity measures are called association measures and take the number of observed cooccurrences into account. This allows to detect similarities in cases where string or numerical similarity measures can not detect any similarity, e.g., for detecting synonyms.

Limitations The whole approach, blocking and matching, assumes structured data that confirms to some kind of schema. That means in cases where only some kind of differently structured strings are used as input data, e.g., citations in publications, this approach can not be used directly. In such cases necessary preprocessing steps, like parsing and schema integration, need to be performed beforehand .

Furthermore, the approach to extract representative non-duplicates for the matching phase, assumes that the individual data sources are more or less duplicate free. This also implicitly requires that the duplicates should be detected between two data source in contrast to finding duplicates within a single data source. In practice this is a valid assumption for a large family of applications, e.g., in commercial settings like webshops or product catalogs the data is of such high quality. If these properties do not hold, then the representative non-duplicates must be generated differently, e.g., by manually selecting such pairs. However, the blocking approach does not have

this limitation, i.e., it can also be used directly to find candidate duplicates within a single data source.

Although the main parts of the approach are unsupervised, i.e., they require no human interaction, some modules need to be well parameterized to work effectively. This is the blocking module, where the used keys, the window sizes and the used distance function need to be selected. Furthermore, the comparison module needs to parameterized, which attribute-pairs are compared with which similarity measure or combinations of similarity measures.

In a concrete application not only a ranking based on a duplicate estimation score is relevant, but some concrete decision based on this score, which are declared as duplicates and which are declared as non-duplicates. Some application might even require to distinguish between duplicates, non-duplicates and possible duplicates that are scheduled for further clerical review. The proposed decision models provide some kind of natural threshold for this decision, but based on the requirements of an application, this threshold might not be ideal.

7.2 Future Work

Similarity Measures The experiments using different similarity measures have shown the enormous impact of them on the final duplicate decision. This demands for further work in unsupervised learning of ideal similarity measures. To this end, the result of an initial duplicate decision process could be used to examine, which similarity measures on which attributes have performed well, to separate duplicates from non-duplicates. In this context it is also interesting to examine, which attributes need to be compared at all, i.e., which attributes really contribute to the duplicate decision.

Decision Models The selection of machine learning methods in this work was very limited, but will be broaden in future work. Several recent methods are very promising. First more complex unsupervised clustering methods could be used, these are e.g. spectral clustering [71] and kernel KMeans [69]. Then several methods from the currently active research area of semi-supervised learning [84], which try to make use of only some kind of additional knowledge that is in this case examples for only one of the classes. These semi-supervised methods are e.g. clustering with constraints [4] or what is known as transductive support vector machines (TSVM) [15].

Quality Estimation It would be also very interesting to have a quality estimation for the final duplicate decision process, i.e., some kind of confidence in the results. If this confidence is low, it would be also interesting to know, what is reason for that. This can be poorly performing similarity measures or simply insufficient data for a good duplicate detection.

Other Other interesting future research directions are incorporating duplicate detection experiences with other data sources into the process of integrating new data sources and duplicate detection without the assumption of a fully successful schema integration in advance.

List of Publications

This work was published in the following articles:

S. Agarwal, P. Fankhauser, J. Gonzalez-Ollala, J. Hartmann, S. Hollfelder, A. Jameson, S. Klink, P. Lehti, M. Ley, E. Rabbidge, E. Schwarzkopf, N. Shrestha, N. Stojanovic, R. Studer, G. Stumme, B. Walter, and A. Weber. Semantic Methods and Tools for Information Portals. In *INFORMATIK 2003 - Innovative Informatikanwendungen, Band 1, Beiträge der 33. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 29. September - 2. Oktober 2003 in Frankfurt am Main*, volume 34 of *LNI*, pages 116–131, 2003.

P. Fankhauser, N. Fuhr, J. Hartmann, A. Jameson, C.-P. Klas, S. Klink, A. Koschmider, S. Kriewel, P. Lehti, P. L. 0002, E. W. Mayr, A. Oberweis, P. Ortyl, S. Pfingstl, P. Reuther, U. Rusnak, G. Sautter, K. Böhm, A. Schaefer, L. Schmidt-Thieme, E. Schwarzkopf, N. Stojanovic, R. Studer, R. Vollmar, B. Walter, and A. Weber. Fachinformationssystem Informatik (fis-i) und Semantische Technologien für Informationsportale (semiport). In *INFORMATIK 2005 - Informatik LIVE! Band 2, Beiträge der 35. Jahrestagung der Gesellschaft für Informatik e.V. (GI), Bonn, 19. bis 22. September 2005*, volume 68 of *LNI*, pages 698–712, 2005.

P. Lehti and P. Fankhauser. A Precise Blocking Method for Record Linkage. In *Proceedings of the 7th International Conference on Data Warehousing and Knowledge Discovery (DaWaK'05)*, pages 210–220, Aug. 2005.

P. Lehti and P. Fankhauser. Probabilistic Iterative Duplicate Detection. In *Proceedings of the International Conference on Ontologies, Databases and Applications of Semantics at On the Move to Meaningful Internet Systems 2005 (ODBASE'05)*, pages 1225–1242, Nov. 2005.

P. Lehti and P. Fankhauser. Unsupervised Duplicate Detection using Sample Non-Duplicates. *Journal on Data Semantics (JoDS VII), Normal Issue, LNCS 4244*, to appear in Automn 2006.

P. Lehti, P. Fankhauser, S. von Stackelberg, and N. Shrestha. XML-based Data Integration for Semantic Information Portals. In *INFORMATIK 2004 - Informatik verbindet, Band 2, Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), Ulm, 20.-24. September 2004*, volume 51 of *LNI*, pages 159–163. GI, 2004.

List of Figures

List of Tables

Bibliography

[1] AIFB-Karlsruhe. Semantic Methods and Tools for Information Portals. http://km.aifb.uni-karlsruhe.de/projects/semiport/.

[2] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. Eliminating fuzzy duplicates in data warehouses. In *Proceedings of the 28th International Conference on Very Large Data Bases(VLDB '02)*, 2002.

[3] R. Baeza-Yates and B. Ribiero-Neto. *Modern Information Retrieval*, pages 74–79. Addison Wesley, 1999.

[4] S. Basu, M. Bilenko, and R. J. Mooney. A probabilistic framework for semi-supervised clustering. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004)*, Aug. 2004.

[5] R. Baxter, P. Christen, and T. Churches. A comparison of fast blocking methods for record linkage. In *Proceedings of the Workshop on Data Cleaning, Record Linkage and Object Consolidation at the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, aug 2003.

[6] T. R. Belin and D. B. Rubin. A method for calibrating false-match rates in record linkage. *Journal of the American Statistical Association*, 90(430):694–707, 1995.

[7] I. Bhattacharya and L. Getoor. Deduplication and group detection using links. In *Proceedings of the KDD-2004 Workshop on Link Analysis and Group Detection*, Aug. 2004.

[8] I. Bhattacharya and L. Getoor. Iterative record linkage for cleaning and integration. In *Proceedings of the SIGMOD 2004 Workshop on Research Issues on Data Mining and Knowledge Discovery*, June 2004.

[9] M. Bilenko, S. Basu, and M. Sahami. Adaptive product normalization: Using online learning for record linkage in comparison shopping. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM-2005)*, 2005.

[10] M. Bilenko and R. J. Mooney. Learning to combine trained distance metrics for duplicate detection in databases. Technical Report AI 02-296, Artificial Intelligence Laboratory, University of Texas at Austin, Austin, TX, Feb. 2002.

[11] M. Bilenko and R. J. 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)*, Washington, DC, 2003.

[12] M. Bilenko and R. J. Mooney. Employing trainable string similarity metrics for information integration. In *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web*, pages 67–72, Acapulco, Mexico, Aug. 2003.

[13] A. Borthwick, M. Buechi, and A. Goldberg. Key concepts in the choicemaker 2 record matching system. In *Proceedings of the 2003 ACM SIGKDD Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, pages 28–30, Washington, DC, 2003.

[14] C.-C. Chang and C.-J. Lin. Libsvm - a library for support vector machines. http://www.csie.ntu.edu.tw/ cjlin/libsvm/.

[15] O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 57–64, 2005.

[16] S. Chaudhuri, V. Ganti, and R. Motwani. Robust identification of fuzzy duplicates. In *Proceedings of the 21st International Conference on Data Engineering (ICDE-2005)*, Tokyo, Japan, 2005.

[17] P. Christen and T. Churches. Febrl: Freely extensible biomedical record linkage manual, release 0.2 edition. https://sourceforge.net/projects/febrl/, 2003.

[18] K. W. Church, W. Gale, P. Hanks, and D. Hindle. Using statistics in lexical analysis. *Lexical Acquisition: Using On-line Recources to Build a Lexicon*, pages 115–164, 1991.

[19] K. W. Church and P. Hanks. Word association norms, mutual information and lexicography. *Computational Linguistics*, 16(1):22–29, 1990.

[20] W. W. Cohen. Integration of heterogeneous databases without common domains using queries based on textual similarity. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data (SIGMOD-98)*, pages 201–212, 1998.

[21] W. W. Cohen, H. Kautz, and D. McAllester. Hardening soft information sources. In *Proceedings of the 6th International Conference on Knowledge Discovery and Data Mining (KDD-2000)*, pages 255–259, Boston, MA, August 2000.

[22] W. W. Cohen, P. Ravikumar, and S. Fienberg. Secondstring - an open-source java-based package of approximate string-matching techniques. http://secondstring.sourceforge.net/.

[23] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string metrics for matching names and records. In *Proceedings of the KDD-2003 Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, pages 13–18, Washington, DC, 2003.

[24] W. W. Cohen and J. Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002)*, Edmonton, Alberta, 2002.

[25] A. Culotta and A. McCallum. Joint deduplication of multiple record types in relational data. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 257–258, New York, NY, USA, 2005. ACM Press.

[26] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 34:1–38, 1977.

[27] A. Doan, Y. Lu, Y. Lee, and J. Han. Object matching for information integration: A profiler-based approach. In *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web*, pages 53–58, Acapulco, Mexico, Aug. 2003.

[28] X. Dong, A. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *Proceedings of the 2005 ACM SIGMOD in-*

ternational conference on Management of data (SIGMOD-2005), pages 85–96, Baltimore, MD, 2005.

[29] M. G. Elfeky, V. S. Verykios, and A. K. Elmargarid. Tailor: A record linkage toolbox. In *Proceedings of the 18th International Conference on Data Engineering (ICDE'02)*, page 17, Washington, DC, USA, 2002. IEEE Computer Society.

[30] S. Evert. Computational approaches to collocations. http://www.collocations.de/.

[31] Fachinformationszentrum-Karlsruhe. CompuScience. http://www.zblmath.fiz-karlsruhe.de/cs/.

[32] C. Faloutsos and K.-I. Lin. FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 163–174, San Jose, California, 1995.

[33] I. P. Fellegi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64:1183–1210, 1969.

[34] H. Galhardas, D. Florescu, D. Shasha, and E. Simon. An extensible framework for data cleaning. In *Proceddings of the 16th International Conference on Data Engineering (ICDE '00)*, page 312, 2000.

[35] H. Galhardas, D. Florescu, D. Shasha, E. Simon, and C. Saita. Declarative data cleansing: Language, model, and algorithms. In *Proceedings of the 27th International Conference on Very Large Databases (VLDB-2001)*, pages 371–380, Rome, Italy, 2001.

[36] L. Gravano, P. G. Ipeirotis, H. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava. Approximate string joins in a database (almost) for free. In *Proceedings of the 27th VLDB Conference (VLDB'01)*, pages 491–500, Rome, Italy, 2001.

[37] L. Gu and R. Baxter. Adaptive filtering for efficient record linkage. In *Proceedings of the Fourth SIAM International Conference on Data Mining*, Apr. 2004.

[38] M. A. Hernandez and S. J. Stolfo. Real-world data is dirty: Data cleansing and the merge/purge problem. *Data Mining and Knowledge Discovery*, 2(1):9–37, 1998.

[39] S. Hill. Social network relational vectors for anonymous identity matching. In *Proceedings of the IJCAI-2003 Workshop on Learning Statistical Models from Relational Data*, pages 48–52, Acapulco, Mexico, Aug. 2003.

[40] J. A. Hylton. Identifying and merging related bibliographic records. Master's thesis, Department of Electrical Engineering and Computer Science, MIT, 1996.

[41] M. Jaro. Advances in record linkage methodology as applied to matching the 1985 census of tampa. *Journal of the American Statistical Society*, 84:414–420, 1989.

[42] M. Jaro. Software demonstrations. In *Proceedings of an International Workshop and Exposition - Record Linkage Techniques*, Arlington, VA, 1997.

[43] L. Jin, C. Li, and S. Mehrotra. Efficient record linkage in large data sets. In *Proceedings of Int. Conf. on Database Systems for Advanced Applications (DASFAA)*, Tokyo, Japan, March 2003.

[44] D. V. Kalashnikov, S. Mehrotra, and Z. Chen. Exploiting relationships for domain-independent data cleaning. In *Proceedings of the 5th SIAM International Conference on Data Mining (SDM-2005)*, Newport Beach, CA, 2005.

[45] J. Kang, D. Lee, and P. Mitra. Identifying value mappings for data integration: An unsupervised approach. In *Proceedings of the 14th International Conference on Web Information Systems Engineering (WISE-2005)*, New York, NY, USA, Nov. 2005.

[46] R. P. Kelley. Advances in record linkage methodology: a method for determining the best blocking strategy. In *Proc. of the Workshop on Exact Matching Methodologies in Arlington - Record Linkage Techniques*, 1985.

[47] M. D. Larsen and D. B. Rubin. Alternative automated record linkage using mixture models. *Journal of the American Statistical Association*, 79:32–41, 2001.

[48] M.-L. Lee, T. W. Ling, and W. L. Low. Intelliclean: a knowledge-based intelligent data cleaner. In *Proceedings of the Sixth International Conference On Knowledge Discovery and Data Mining (KDD-2000)*, pages 290–294, Boston, MA, 2000.

[49] M. L. Lee, H. Lu, T. W. Ling, and Y. T. Ko. Cleansing data for mining and warehousing. In *Proceedings of the 10th International Conference on Database and Expert Systems Applications (DEXA-99)*, Florence, Italy, August 1999.

[50] V. I. Levenshtein. Binary codes capable of correcting insertions and reversals. *Soviet Physics Doklady*, 10:707–710, February 1966.

[51] M. Ley. DBLP computer science bibliography. http://dblp.uni-trier.de/.

[52] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Math., Stat. and Prob.*, pages 281–296, 1967.

[53] A. McCallum and B. Wellner. Toward conditional models of identity uncertainty with application to proper noun coreference. In *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web*, pages 79–86, Acapulco, Mexico, Aug. 2003.

[54] A. McCallum and B. Wellner. Conditional models of identity uncertainty with application to noun coreference. In *Advances in Neural Information Processing Systems 17*, pages 905–912. MIT Press, 2005.

[55] A. K. McCallum, K. Nigam, and L. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the 6th International Conference On Knowledge Discovery and Data Mining (KDD-2000)*, pages 169–178, Boston, MA, Aug. 2000.

[56] M. Michalowski, S. Thakkar, and C. A. Knoblock. Exploiting secondary sources for automatic object consolidation. In *Proceedings of the KDD-2003 Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, Washington, DC, 2003.

[57] M. Michalowski, S. Thakkar, and C. A. Knoblock. Exploiting secondary sources for unsupervised record linkage. In *Proceedings of the 30th International Conference on Very Large Data Bases(VLDB '04)*, 2004.

[58] A. Monge and C. Elkan. An efficient domain independent algorithm for detecting approximately duplicate database records. In *In Proceedings of the SIGMOD Workshop on Data Mining and Knowledge Discovery*, 1997.

[59] A. E. Monge and C. P. Elkan. The field matching problem: Algorithms and applications. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 267–270, Portland, OR, August 1996.

[60] M. Neiling and S. Jurk. The object identification framework. In *Proceedings of the 2003 ACM SIGKDD Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, pages 37–39, Washington, DC, 2003.

[61] H. B. Newcombe, J. M. Kennedy, S. J. Axford, and A. P. James. Automatic linkage of vital records. *Science*, 130:954–959, 1959.

[62] G. N. Norén, R. Orre, and A. Bate. A hit-miss model for duplicate detection in the WHO Drug Safety Database. In *Proceedings of the 11th International Conference on Knowledge Discovery and Data Mining (KDD-05)*, pages 459–468, Chicago, IL, 2005.

[63] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. Identity uncertainty and citation matching. In *Advances in Neural Information Processing Systems 15*. MIT Press, 2003.

[64] K. Pelckmans, J. D. Brabanter, J. Suykens, and B. D. Moor. Handling missing values in support vector machine classifiers. *Neural Networks*, 18(5–6):684–692, 2005.

[65] P. Ravikumar and W. W. Cohen. A hierarchical graphical model for record linkage. In *AUAI '04: Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 454–461. AUAI Press, 2004.

[66] E. S. Ristad and P. N. Yianilos. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):522–532, 1998.

[67] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2002.

[68] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002)*, Edmonton, Alberta, 2002.

[69] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[70] W. Shen, X. Li, and A. Doan. Constraint-based entity matching. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-2005)*, pages 862–867, Pittsburgh, PA, 2005.

[71] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. CAD-Integrated Circuits and Systems*, 13:888–905, Aug. 2000.

[72] P. Singla and P. Domingos. Multi-relational record linkage. In *Proceedings of the KDD-2004 Workshop on Multi-Relational Data Mining*, pages 31–48, Aug. 2004.

[73] P. Singla and P. Domingos. Object identification with attribute-mediated dependences. In *Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases (PAKDD-2005)*, Porto, Portugal, 2005.

[74] S. Y. Sung, Z. Li, and P. Sun. A fast filtering scheme for large database cleansing. In *Proceedings of the (CIKM'02)*, McLean, Virginia, November 2002.

[75] S. Tejada, C. A. Knoblock, and S. Minton. Learning object identification rules for information integration. *Information Systems Journal*, 26(8):635–656, 2001.

[76] S. Tejada, C. A. Knoblock, and S. Minton. Learning domain-independent string transformation weights for high accuracy object identification. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002)*, Edmonton, Alberta, 2002.

[77] V. S. Verykios, G. V. Moustakides, and M. G. Elfeky. A bayesian decision model for cost optimal record matching. *The VLDB Journal*, 12(1):28–40, 2003.

[78] B. Wellner, A. McCallum, F. Peng, and M. Hay. An integrated, conditional model of information extraction and coreference with application to citation matching. In *Proceedings of the 2004 Conference on Uncertainty in Artificial Intelligence (UAI-2004)*, July 2004.

[79] W. E. Winkler. Using the em algorithm for weight computation in the fellegi-sunter model of record linkage. In *Proceedings of the Section on Survey Research Methods, American Statistical Association*, pages 667–671, 1988.

[80] W. E. Winkler. Improved decision rules in the fellegi-sunter model of record linkage. In *Proceedings of the Section on Survey Research Methods, American Statistical Association*, pages 274–279, 1993.

[81] W. E. Winkler. The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Census Bureau, Washington, DC, 1999.

[82] W. E. Winkler. Data cleaning methods. In *Proceedings of the KDD-2003 Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, pages 1–6, Washington, DC, 2003.

[83] W. E. Winkler. Approximate string comparator search strategies for very large administrative lists. Technical report, Statistical Research Division, U.S. Census Bureau, Washington, DC, 2005.

[84] X. Zhu. *Semi-Supervised Learning with Graphs*. PhD thesis, Carnegie Mellon University, 2005. CMU-LTI-05-192.