Key Concepts in the ChoiceMaker 2 Record Matching System

Andrew Borthwick
ChoiceMaker Technologies, Inc.
646-336-4442
aborthwick@choicemaker.com

Martin Buechi
ChoiceMaker Technologies, Inc.
71 W. 23rd St., Suite 515
New York, NY 10010
646-336-4443
mbuechi@choicemaker.com

Arthur Goldberg
ChoiceMaker Technologies, Inc. and
New York University
Computer Science Department
251 Mercer, Room 409
New York, NY 10012
212-998-3014
agoldberg@choicemaker.com

ABSTRACT
We describe an innovative record matching system called ChoiceMaker 2 we developed at ChoiceMaker Technologies (CMT). Firstly, we describe the process by which we use a machine learning technique known as maximum entropy modeling to tune the system to the problem at hand. Secondly, we describe the ClueMaker™ programming language that is used to describe record matching characteristics. Thirdly, we describe our method for testing record matching systems and describe how our IDE facilitates this process.

Categories and Subject Descriptors
D.3.3 [**Data Quality**]:

General Terms
Algorithms, Standardization, Languages, Verification.

Keywords
Record matching, record linkage, merge/purge, object consolidation, data quality, maximum entropy, machine learning, Java, artificial intelligence, deduplication.

1. INTRODUCTION
Approximate record matching is needed in the absence of an accurate, always available, unique key. Record-matching tasks can be broken down into three main categories:

- Duplicate record removal or linkage: The same person, business, or thing is present more than once in a database. Duplicate records are identified and then linked together, merged or one is removed (purged).
- Database linkage: Two databases are linked or merged.
- Approximate database search: A database is searched for records similar to an input record.

In all of these cases, the core problem is essentially the same: given a query record, try to find in a target database the record(s) that denote the same thing (e.g., a person or company) as the query record.

1.1 Matching Process Overview
ChoiceMaker 2 performs record matching as a two-step process. This process is illustrated in Figure 1:

1. *Query record.* A query record is input.
2. *Blocking*. The matching engine searches the target database for records that are possible matches to the query record. The objective at this stage is to retrieve all possible matches and not too many non-matches.
3. *Scoring*. The matching engine determines for each possible match the probability that it denotes the same thing as the query record. Possible matches are categorized into matches, potential matches, and non-matches based on two user-defined thresholds. For instance, any record matching the query record with a probability higher than the "match threshold" is declared a "match".

In this paper, we will focus on the scoring stage.

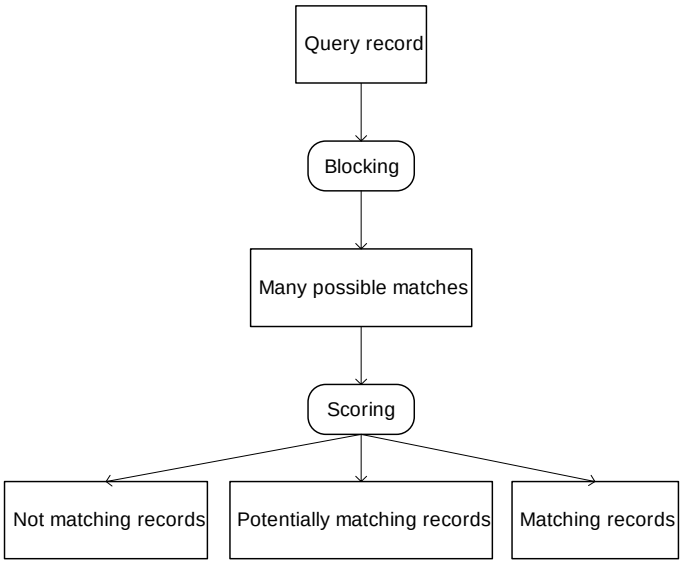

Figure 1: ChoiceMaker Matching Overview

2. MACHINE LEARNING

ChoiceMaker's record matching models are built around a set of "clues" (commonly known in the AI literature as "features"), which indicate whether a pair of records "match" or "differ". Clues can be arbitrary predicates of the record-pair. Some sample clues for a database of people include:

- Are the first names the same and are they common, uncommon, or rare?
- Do the last names have the same phoneticization according to Soundex [3] or similar techniques?
- Is the date of birth different?

These clues are written in ChoiceMaker's ClueMaker™ programming language described in section 3.

Our Machine Learning (ML) approach constructs a record matching model that outputs the probability that a pair of records represents the same entity. The model compares the input record with each possibly matching record. The model is trained on a set of pairs of records that have been tagged as a "match", "differ", or "hold" (unsure). Although we have experimented with other ML techniques, ChoiceMaker currently uses maximum entropy modeling (ME) [1] [2] in its production models. Each clue predicts a decision (a "future" in the AI literature) of "match" and "differ". The ME training process then assigns to each clue a weight, which is a positive real number indicating the relative predictive strength of the clue. In ME, at run time the weights of all active clues form the probability of match:

$$\text{Probability} = \frac{\text{MatchProduct}}{\text{MatchProduct} + \text{DifferProduct}}$$

where MatchProduct is defined as the product of the weights of all clues predicting "match" for the pair and DifferProduct is the product of the weights of all clues predicting "differ" for the pair.

Relative to other machine learning techniques, ME has proven to be a good choice for this problem because it is fast at run time, is relatively easy to explain to clients, and can assign accurate weights to multiple overlapping and interacting clues.

3. CLUEMAKER™ PROG. LANGUAGE

3.1 Why a new programming language?

The introduction of a new programming language requires strong justification as IT departments are understandably hesitant to support another language. We created ClueMaker for the following reasons:

- A set of clues written in ClueMaker is roughly ten times shorter than the same set of clues written in Java.
- Clues written in ClueMaker are more easily understood by customers.
- Since ClueMaker contains many constructs specific to record matching it is less error prone than repetitious boilerplate code in Java.
- ClueMaker allows for many code optimizations (such as common subexpression elimination and code invariant loop motion) that cannot be applied to Java programs because of side effects. The performance gain of these optimizations can often reach a factor of ten.

3.2 ChoiceMaker Schemas

The entities (records) that are to be matched are described by a ChoiceMaker 'schema', which takes the form of an XML document. The schema defines the name and type of each field. It also includes validity predicates which define the valid values for each field. For instance, a first name of "Boy" is not valid, and a social security number shouldn't begin with "000". Finally, ChoiceMaker schemas can define derived values, so, for example, we can produce fields such as "city", "state", and "zip code" from an incoming unparsed address.

3.3 ClueMaker and Java

ClueMaker adds a thin layer of syntax tailored to the record matching problem on top of Java. The ClueMaker compiler compiles the ClueMaker code into Java source code, after which a Java compiler compiles it into byte code.

Most of ClueMaker consists of Java expressions. For example, the first expression in Figure 2 defines the clue "mFirstName". The keyword **match** indicates that the clue predicts "match". The keyword **valid** takes a field-name argument and refers to the corresponding validity definition of the schema. This is checked on the "firstName" field of the two records being compared, which are always referred to as "q" (the input, or "query" record) and "m" (the possible match record retrieved from the database).

The expression "q.firstName == m.firstName" is simply a Java expression. These expressions can be arbitrary Java. For instance, in dLastNameSoundex, we see a call to the Java method "soundex" in the Soundex class.

```
clue mFirstName {
      match  valid(q.firstName) && valid(m.firstName)
                && q.firstName == m.firstName;
}

clue dLastNameSoundex {
      differ valid(q.lastName) && valid(m.lastName)  &&
            Soundex.soundex(q.lastName) !=
            Soundex.soundex(m.lastName);
}
```

Figure 2: Sample ClueMaker code

Consequently, ClueMaker is easily learned by Java programmers. The core of each clue consists of actual Java code supplemented by various ClueMaker constructs. Library methods such as "Soundex" can be defined in standard Java. The ClueMaker language manual devotes just 25 pages to describing the novel features of the language.

3.4 ClueMaker Features

ClueMaker contains innovative features which greatly aid development of a matching model. These include:

- The shorthand forms "same" and "different" which simultaneously check validity and whether the fields in question are the same or not.

- A “swap” construct which checks for fields being swapped (e.g., first name in the last name field).
- An ability to work with “stacked data”, schemas which permit multiple values for the same field. For instance, many databases allow multiple values for address, names, etc.

The talk will include sample code from all of these different constructs.

The enormous productivity gain that ChoiceMaker has seen from the introduction of the ClueMaker programming language has more than justified the project.

4. TESTING

After developing clues in ClueMaker for our record matching model and training it on hand-marked data using ME, we are ready to test the model on a separate corpus of hand-marked data on which the model has not been trained and which CMT personnel have not seen. This testing on unseen data is performed on data tagged “match”, “differ” or “unsure” by the client.

ChoiceMaker recommends that its clients evaluate the system by first determining the false-positive and false-negative levels of accuracy that they are comfortable with, where a false-positive is a record-pair identified as a match by ChoiceMaker which is human tagged as “differ” and a false-negative is a pair human tagged as “match” which ChoiceMaker identifies as “differ”. Different clients may have different relative tolerances for false-positives and false-negatives. For instance, in a medical context, falsely identifying two individuals as being the same person could be disastrous. On the other hand, in a counter-terrorism situation, it may be more important not to miss any possibly matching suspects, even at the expense of falsely identifying non-terrorists as requiring investigative follow-up.

Given the client’s stated tolerance for false-positive and false-negative responses, we then test the model in our ModelMaker tool, which determines the “match” and “differ” thresholds (see section 1.1) that will yield no more than the desired percentage of errors. The model is then judged on the percentage of record pairs that fall between the match and differ thresholds. Since these records will need to be evaluated by hand as to whether they are matches or not, the model is judged on how low this percentage is.

This portion of the talk will include a demonstration of ChoiceMaker’s ModelMaker IDE, focusing on a suite of screens which facilitate the evaluation of record matching models.

5. ACKNOWLEDGMENTS

This work was supported, in part, by National Science Foundation SBIR grants DMI-0060675 and DMI-0216213. Adam Winkel contributed to the design of ClueMaker. Members of the NYU Programming Language and Database seminars provided valuable feedback on an early draft of ClueMaker.

6. REFERENCES

[1] Berger, A., Della Pietra, S. A., and Della Pietra, V. J., A Maximum Entropy Approach to Natural Language Processing *Computational Linguistics*, vol. 22, pp. 39-71, 1996.

[2] Borthwick, A., A Maximum Entropy Approach to Named Entity Recognition 1999. New York University.

[3] Knuth, D. *The Art of Computer Programming*, Addison-Wesley, 1998.