Cleaning and Visualizing a dirty set of restaurant data

Florian Loher
Technical University of Applied Science Regensburg
florian.loher@st.oth-regensburg.de

***Abstract*—Duplicate detection is an important part of data cleaning. Many approaches use string matching as a means to calculate the similarity of records. In this article I elaborate on the use of semi-random training sets combined with the SoftTF-IDF similarity measure, reaching precision and recall values over 85% for relatively small training sets.**

***Index Terms*—Data cleaning, duplicate detection**

I. Introduction

Big data is a rapidly growing field of research that already gained overwhelming interest in the general public. The amount of data is increasing at an exponential rate and is likely to grow further at this rate. To be able to leverage the power of data, the need for ways to clean it is as high as ever.

Data cleaning, also referred to as data scrubbing or data cleansing, is a research field concerned with improving the quality of faulty data. Typical aspects that are sought to be improved are the amount of duplicates, type errors or inconsistencies[1].

In this article I am going to outline a possible approach to detecting duplicates in a dataset of 864 restaurants[1]. I first audit the data. Then I generate a training set, standardize fields and choose *SoftTF-IDF* string matching measure with *Jaro-Distance* as sub measure as the algorithm for duplicate detection. After creating a gold standard for the training set I train the thresholds for restaurant name, phone number and address similarity that determine if two records will be declared duplicates. Lastly the detection algorithm is run on the test data and compared to the gold standard[2]. Using and comparing different sizes of training data I show that even with a training set 10% the size of the test data (86 records, 22 of which are part of duplicates) a recall of ∼86% with a precision of ∼95% is typically achieved.

In section II I am going to introduce basic terminology concerning duplicate detection and describe the string-matching algorithms I use. Section III will cover the process of processing and standardizing the data to enhance the results of duplicate detection. The reasoning behind the manner of training data generation is explained in section IV. Then in section V I will reason for the use of the string-matching algorithm SoftTF-IDF. Section VI describes the methodology of training and testing followed by the presentation and comparison of my results in section VII. Finally I will draw conclusions in section VIII and give an outlook to possible future research as well as point to some related work.

II. Basics of duplicate detection

A major field of research is duplicate detection i.e. trying to find methods that recognize if two distinct records in a given data set actually represent the same real world entity. Such pairs of records are called duplicates.

The degree to which an algorithm, method or process is able to detect duplicates can be measured if a gold standard addressing the relevant data set is available. A gold standard is a complete list of every duplicate that exists in the data.

The way of comparing the algorithm to the gold standard is to calculate its precision and recall. Precision is defined as $\frac{|TP|}{|TP|+|FP|}$ where $|TP|$ is the number of records both the algorithm and the gold standard recognize as duplicates (true positives) and $|FP|$ is the number of records the algorithm recognizes as duplicates but the gold standard does not (false positives). Recall is defined as $\frac{|TP|}{|TP|+|FN|}$ where $|TP|$ is the same as before and $|FN|$ is the number of true duplicates the algorithm does not find (false negatives).

A. String matching

Duplicate detection relies on the matching of strings i.e. comparing fields of distinct records in a given dataset and calculating how similar they are, based on a chosen measure.

Token-based measures are typically optimized to handle differences where substrings are swapped or rotated. Frequent differences between fields that represent the same entity are for example swapping first and last name, or having a title prepended or put at the end. By splitting the strings into tokens and comparing the resulting sets of strings these rotations cease to impact those measures.

WHIRL, a token-based similarity measure proposed by [4] additionally utilizes the *TF-IDF* (term frequency, indirect document frequency) weighing scheme. This also

[1]Restaurant data provided at https://hpi.de/fileadmin/user_upload/fachgebiete/naumann/projekte/repeatability/Restaurants/restaurants.tsv

[2]Gold standard provided at https://hpi.de/fileadmin/user_upload/fachgebiete/naumann/projekte/repeatability/Restaurants/restaurants_DPL.tsv

prioritizes terms that appear rarely in the document giving credit to the thought that strings are probably semantically equivalent if they share a lot of defining terms. Frequent terms, such as *street* in a field that represents an address, however are less likely to point to duplicates and should be valued less.

A further subtlety is introduced by [1] with the measure *SoftTF-IDF*. Instead of needing exact matches for the tokens like WHIRL, SoftTF-IDF utilizes a character-based similarity measure to calculate similarity values for tokens that are not exact matches.

Character-based similarity measures often handle typing errors very well and are typically optimized for specific types of strings such as names. The *Jaro-Distance* metric introduced by [5] that is intended for first and last names. It tries to match strings by finding common characters and calculating the number of transpositions by counting the number of non-matching common characters at equal positions.

III. Auditing and data preparation

Before trying to find duplicate data in any given dataset most data cleaning approaches start with a phase of preparing the data. Data preparation usually contains the steps parsing, transformation and standardization. This includes but is by far not limited to discovering which types of fields are present in the data and removing unnecessary characters from fields.

In the case of the restaurants data I am looking at, a quick audit reveals that entries in the "phone" fields do not conform to a common format. The phone field of record 97 is *212/627-8273* while record 98 contains the value *212-627-8273*. The separation characters between numbers are not uniform. This can be observed throughout the entirety of the data set. A non extensive record of further inconsistencies can be found in table I.

In the parsing and transformation phase I copy the data to an instance of *MongoDB*[3] in order to make the data easier accessible in the following stages. Each row of the original data is transformed into a record in the database. In MongoDB the equivalent to tables of relational databases are collections. Each step of my data cleaning process creates a new collection with the updated data.

The process of standardization is split up, targeting each field of the records separately. Because of that the order in which the fields are standardized is irrelevant to the resulting data.

Each field is standardized by the use of different regular expressions that are suited to find the irregularities in the field. Furthermore replacements are done using dictionaries of standardized values. For example the address field contains street types which are sometimes abbreviated. One regular expression is used to find all abbreviations in a field and a dictionary containing street types and their common abbreviations enables the replacement of each instance by its non abbreviated form.

[3]MongoDB is a document database. For more information visit https://www.mongodb.com/

field	type of inconsistency	example
city	– Irregular use of abbreviations	Sometimes *la* instead of *los angeles*
	– Use of city district name as city name	*hollywood* instead of *los angeles*
type	– Irregular use of abbreviations	Sometimes *bbq* instead of *barbecue*
	– Use of different separators for types	*greek and middle eastern* and *russian/german*)
address	– Irregular use of abbreviations	Sometimes *blvd* instead of *boulevard*, sometimes *ninth* instead of *9th*
	– Presence of directions	*60 w. 55th st. between 5th and 6th ave.*
	– Irregular choice of abbreviations	*blvd* or *blv*
phone	– Irregular use of separation characters	Sometimes *212/627-8273* instead of *212-627-8273*

Table I: Types of inconsistencies present in restaurant data

IV. Generation of training data

Since similarity measures typically return a value between 0 and 1 thresholds must be set for which two records are considered semantically equivalent. One way of finding such thresholds is to set them according to prior experience. A similar approach is the use of training data. In this case I am using a subsets of the original data of different sizes. In section VII I will compare the effect the size of the training data has in regards to precision and recall.

The subsets are chosen randomly with the restriction of including duplicates at the same ratio as in the complete data set. This is necessary as random pairs of records are very unlikely to be duplicates. Consider for example a set of 100 records containing 50 pairs of duplicates. Regardless which records are chosen both of them are going to be part of a duplicate pair. The probability that both records are part of the same duplicate pair is $P(x \in D) \cdot P(y \equiv x) = 1 \cdot \frac{1}{99} \approx 1,01\%$, where $P(x \in D)$ it the probability that a randomly chosen record x is part of a duplicate and $P(y \equiv x)$ is the probability that the two randomly chosen records x, y are part of the same duplicate. This calculation shows that a completely random subset is likely to contain a duplicate ration that is considerably lower than the original data. However it is exactly those duplicates that greatly contribute to the value of the ratio. Therefore it is necessary to include at least some duplicates in the training set deliberately.

Although finding all duplicates in a data set is very difficult and most often requires a lot of manual work, finding a small subset of duplicates is comparatively easy since not all pairs of records need to be considered. One way of finding duplicates may be sorting the records and comparing adjacent records. This process is deliberately left out in this work. Instead the provided gold standard is utilized to ensure the same ratio of duplicates in the training set as in the original data.

V. Choice of string-matching algorithm SoftTF-IDF

As described in section II different measures exist for calculating the similarity between fields of records. In order to mitigate the effect of swapped elements in the string a token-based measure is advantageous. Since streets and names often contain substrings that do not majorly impact the similarity of two records, like *street* or *restaurant* the use of a similarity measure that utilizes an approach including the TD-IDF weighing scheme seems appropriate.

To further decrease the risk of terms not matching because of typing errors or irregular elements that were not targeted by the standardization process the comparison between terms should itself be done by a similarity measure. Therefore I propose the use of SoftTF-IDF. Since the tokens themselves usually do not tend to be long, a similarity measure that optimizes for short strings, such as the Jaro-Distance, is appropriate for token matching.

The used SoftTF-IDF measure is implemented by the python library *py_stringmatching*[4].

VI. Training and Testing

Training is done by first calculating similarities for a small subset of all possible pairs of records. First all fields of records are tokenized with an alphanumeric tokenizer, meaning all non alphanumeric characters are interpreted ad separation characters for tokens. These tokens are sorted alphabetically, for the name and address fields. The phone tokens are left as they are. Now the concatenation of tokens is used to sort the records once for each of the aforementioned fields. In every sort iteration each record is compared to its 4 immediate successors using the SoftTF-IDF algorithm described in section V. The similarity values for each compared pair are stored in a dictionary.

After calculation of the similarity values precision and recall are calculated for all possible combinations of thresholds for phone, name and address similarity. The thresholds are limited to 10% intervals meaning they can only take on one of the values $0, 0.1, 0.2, \ldots, 1$. The combination of thresholds that results in the maximum value for $precision \cdot recall$ is considered optimal and therefore chosen as the one that is used for the test data.

For testing all records of the restaurant data go through the same steps as the test data: tokenization, sorting of tokens, sorting of records, calculation of similarity according to SoftTF-IDF. Of course the step of iterating through different thresholds is omitted. Instead the combination resulting from training is applied.

[4]More information on py_stringmatching can be found at https://sites.google.com/site/anhaidgroup/projects/magellan/py_stringmatching

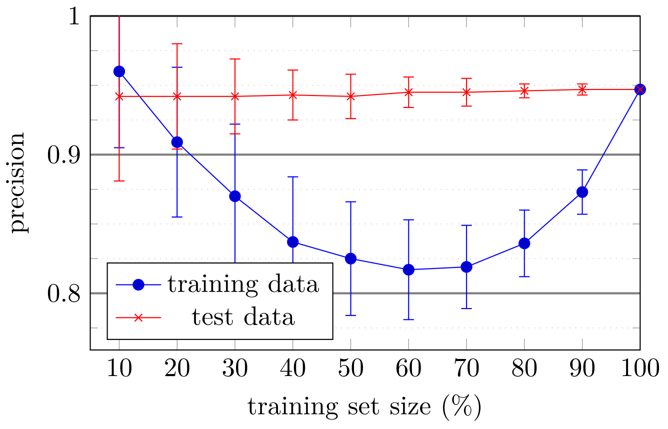


Figure 1: Average precision of duplicate detection in training and test data using different sizes of training sets

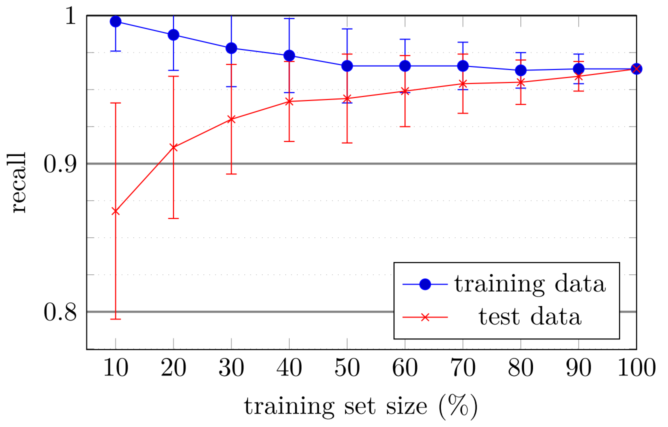


Figure 2: Average recall of duplicate detection in training and test data using different sizes of training sets

The steps of training were done for training data sizes of $10\% - 100\%$ of the restaurant data. The ratios of duplicates were deliberately chosen to be as equal to the original ratio in the data set as possible but to never exceed it. A reference of the exact sizes is provided in appendix A table II. To calculate average precision and recall values for each instance of training set size all instances of training were run 315 times.

VII. Results and comparison of different sizes of training data

Figures 1, 2 and 3 show the results of my testing. They describe the average precision and recall of the training and test sets depending on the training set sizes and the average optimal thresholds for the phone, name and address fields that are used to determine if two values are regarded as duplicates. The results present some interesting findings.

First of all it is noteworthy that even the training sets containing only 86 (~10% of restaurant data) records result in an average recall of ~0.868 with a standard deviation $\sigma \sim 0.073$ combined with an average precision of ~0.942

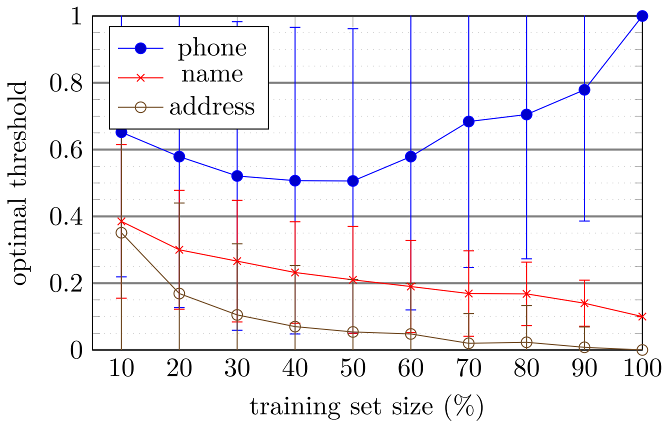


Figure 3: Average optimal thresholds for duplicate detection calculated using different sizes of training sets

with $\sigma \sim 0.061$. While recall increases about 10% to ∼0.964 at 100%, half of this improvement is reached by merely increasing the training set size to 20%.

Precision on the other hand increases only slightly to ∼0.947 resulting in a total difference of only ∼0.005. This could be due to the fact that the precision is already pretty high from the beginning being over 90%. What is more interesting though is that in contrast to test data precision the precision in training is first falling drastically from ∼0.960 at 10% to a minimum average of ∼0.817 at 60% before rising again to 0.947 at 100%. Investigating this unintuitive behaviour in future work could lead to a better understanding of the correlation between precision in training and test data an enable better predictions of likely precision values given a training precision.

Further figures depicting precision and recall in more detail can be found in appendix B.

The optimal thresholds shown in figure 3 present some interesting questions as well. While initially decreasing from ∼0.652 at 10% to ∼0.506 at 50% the threshold of the phone field finally increases to 1.000 at 100% whereas name and address both decrease rather steadily from ∼0.385 to ∼0.100 (name) and from ∼0.351 to 0.000 (address) respectively. The former indicates that the phone field seems to be a good indicator for duplicates if it is used very rigorously. This conforms to the data since most phone numbers of duplicates are identical with only a few outliers. The initial bump however does not lend itself to any immediate conclusions.

The address threshold quickly descending to values near 0 might be an indicator that the chosen similarity measure might not be suitable for address date because intuitively addresses should be a good discriminator for duplicates. An alternative conclusion would be that the threshold of phone entries already encapsulates most of the information that is produced by the address threshold leaving a high address threshold only to declare actual duplicates as non duplicates, which in turn decreases both precision and recall.

VIII. Conclusions, outlook and related work

In this article I showed a possible approach to duplicate detection using the token-based similarity measure SoftTF-IDF and comparing the use of different sized training sets to calculate similarity thresholds. Even small training sets are able to yield presentable results, especially considering the small size of the original data set. Vogel *et al.* [2] describe this as one limitation in the restaurant data set and its gold standard for comparisons regarding certain duplicate detection algorithms. The question if training sets of the same absolute size result in similar precision and recall for larger test data needs further investigation.

Additional research into the causes of the initial bump in phone threshold as well as reasons to why the threshold for the address field diminishes so quickly might prove to reveal further insights. Bilenko *et al.* [3] give a lead to the latter question in proposing that TF-IDF is probably not well suited for detecting duplicates in addresses because of its lack of ability to give different weights to terms that appear similarly often but differ greatly in semantic importance e.g. *34th* a certain street and *place* a simple street suffix. They use the data of restaurants to evaluate the performance of their proposed two-level learning algorithm for duplicate detection, *MARLIN* (Multiply Adaptive Record Linkage with INduction).

References

[1] M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg, "Adaptive name matching in information integration", *IEEE Intelligent Systems*, vol. 18, no. 5, pp. 16–23, 2003, ISSN: 1541-1672. DOI: 10.1109/MIS.2003.1234765.

[2] T. Vogel, A. Heise, U. Draisbach, D. Lange, and F. Naumann, "Reach for gold, An annealing standard to evaluate duplicatedetection results", *Journal of Data and Information Quality*, vol. 5, no. 1-2, pp. 1–25, 2014, ISSN: 19361955. DOI: 10.1145/2629687.

[3] 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*, (Washington, D.C,), T. Senator, Ed., ACM Special Interest Group on Knowledge Discovery in Data and ACM Special Interest Group on Management of Data, New York, NY: ACM, 2003, p. 39, ISBN: 1581137370. DOI: 10.1145/956750.956759.

[4] W. W. Cohen, "Integration of heterogeneous databases without common domains using queries based on textual similarity", *ACM SIGMOD Record*, vol. 27, no. 2, pp. 201–212, 1998, ISSN: 01635808. DOI: 10.1145/276305.276323.

[5] M. A. Jaro and U. M. T. A. United States, "Unimatch, A record linkage system", 1978.

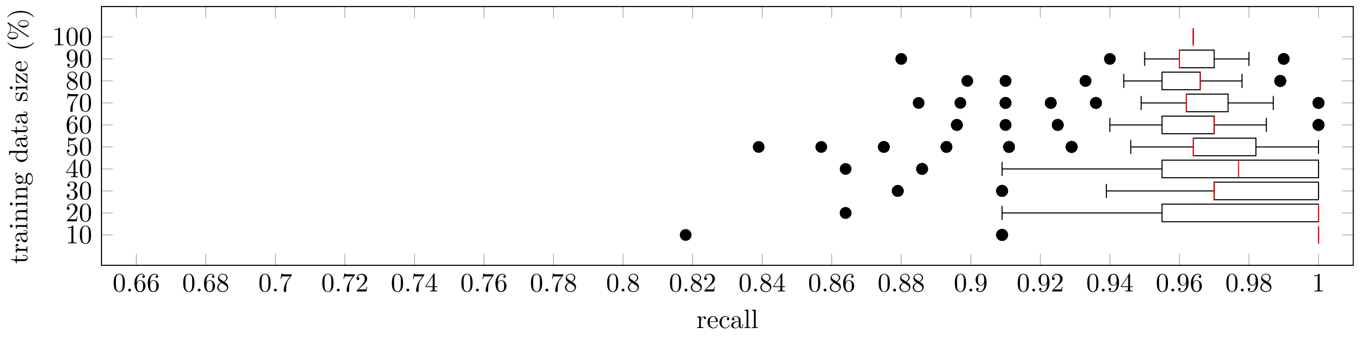
Appendix A
Tables

training size relative (%)	10	20	30	40	50
training size absolute	86	172	259	345	432
number of duplicates	11	22	33	44	56
ratio of duplicates (%)	12,8	12,8	12,7	12,8	13,0

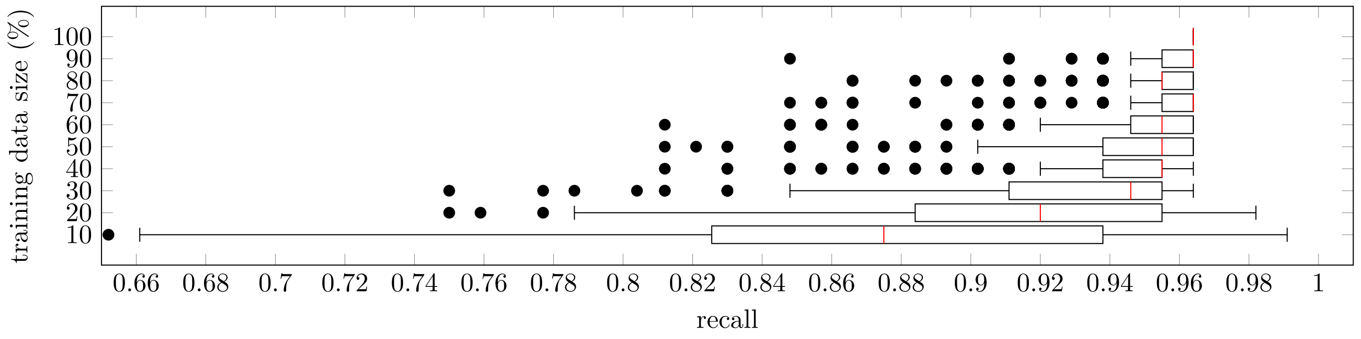
training size relative (%)	60	70	80	90	100
training size absolute	518	604	691	777	864
number of duplicates	67	78	89	100	112
ratio of duplicates (%)	12,9	12,9	12,9	12,9	13,0

Table II: Data sizes and ratios in training sets

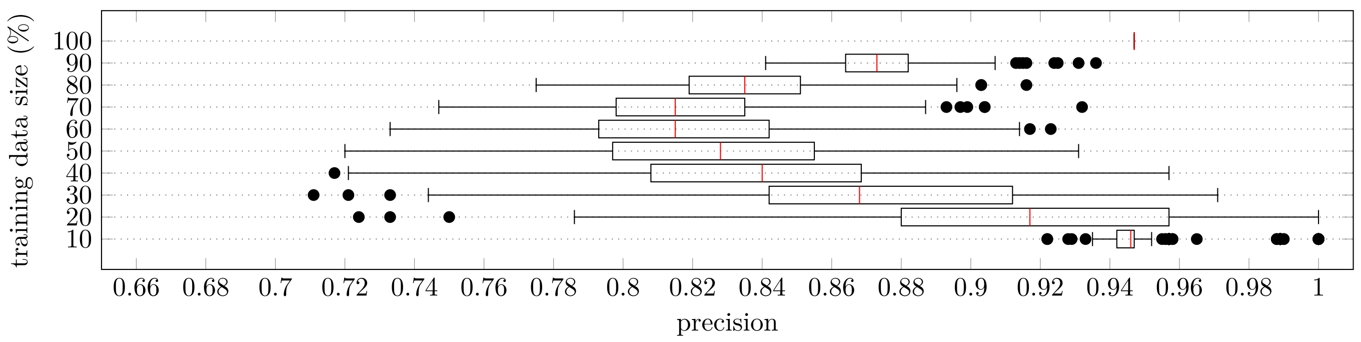
Appendix B
Boxplots of precision and recall for training and test data



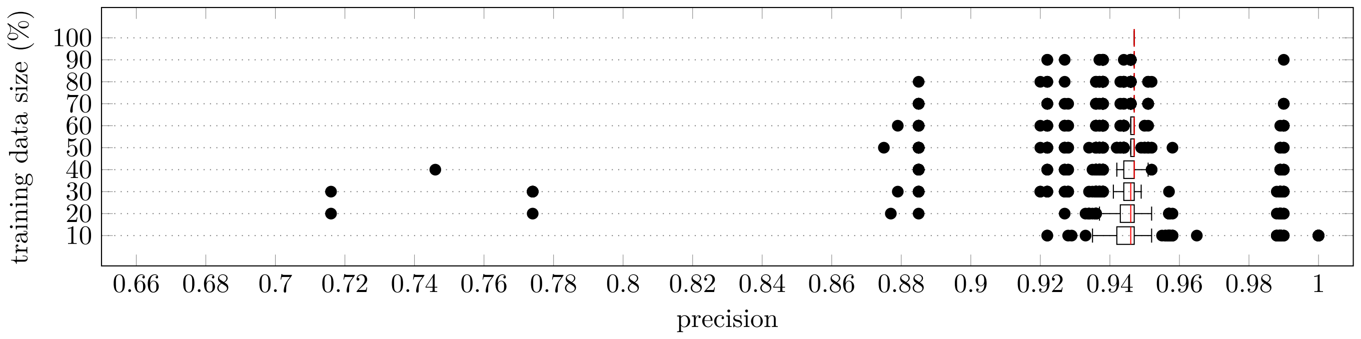
(a) Recall of training data with different sizes of training data and optimal thresholds



(b) Recall of test data with different sizes of training data and optimal thresholds



(a) Precision of training data with different sizes of training data and optimal thresholds



(b) Precision of test data with different sizes of training data and optimal thresholds