Fast and Reliable Acquisition of Truth Data for Document Analysis using Cyclic Suggest Algorithms

Marc-Peter Schambach*
marc-peter.schambach@siemens.com
*Siemens Logistics GmbH
Konstanz, Germany

Stephan von der Nüll*
stephan.vondernuell@siemens.com
*Siemens Logistics GmbH
Konstanz, Germany

Martin Schall*†
martin.schall@htwg-konstanz.de
†Institute for Optical Systems
University of Applied Sciences
Konstanz, Germany

***Abstract*—In document analysis the availability of ground truth data plays a crucial role for the success of a project. This is even more true at the rise of new deep learning methods which heavily rely on the availability of training data. But even for traditional, hand crafted algorithms that are not trained on data, reliable test data is important for the improvement and evaluation of the methods.**

Because ground truth acquisition is expensive and time consuming, semi-automatic methods are introduced which make use of suggestions coming from document analysis systems. The interaction between the human operator and the automatic analysis algorithms is the key to speed up the process while improving the quality of the data. The final confirmation of data may always be done by the human operator.

This paper demonstrates a use case for acquisition of truth data in a mail processing system. It shows why a new, extended view on truth data is necessary in development and engineering of such systems. An overview over the tool and the data handling is given, the advantages in the workflow are shown, and consequences for the construction of analysis algorithms are discussed. It can be shown that the interplay between suggest algorithms and human operator leads to very fast truth data capturing. The surprising finding is the fact that if multiple suggest algorithms circularly depend on data, they are especially effective in terms of speed and accuracy.

I. Introduction

At Siemens Logistics GmbH, machines and systems for the automatic handling and sorting of postal mail, parcels, cargo and air flight baggage are developed. These systems rely on the automatic identification, recognition and interpretation of these items. Data acquisition is done by scanners and cameras for still and video, but also other sensors like laser scanners for barcodes or light curtains for object outline recording are used.

From this data, information is extracted to process the corresponding items, e.g. the destination address, handling information, and item identification. Ground truth data is necessary for algorithmic training, system tuning, evaluation and test during development, but also to define acceptance criteria for the customers.

To meet all these goals, ground truth data has not only to be recorded for the properties on the system's surface, but also for the *intermediate* results of the implementation of the system. These may be the standard algorithmic building steps like foreground/background separation, region of interest detection, image segmentation, raw text recognition, syntactical tagging, etc. This list completely depends on the concrete implementation and may change between projects and over time. Therefore it is necessary to provide a truth data concept, which can be extended to any type of document properties. [1]

Additionally, the scope of truth data in this work goes beyond that of many other document analysis systems and includes structured semantic information. For mail and parcel sorting applications, addresses and distribution codes are the relevant data elements. These are dependent on the actual address databases and coding rules, which must be considered in truth data modeling.

The required amount of truth data has grown significantly over the past years. Basically, there are two main reasons for this: Given the improvements in recognition technology, error rates well below one percent are common and expected. In order to show the significance of accuracy measurements, an appropriate amount of truth data has to be provided. Likewise, with machine learning techniques emerging in many analysis tasks, the need for more training data is given.

Given these needs, the creation of truth data plays an increasing role in document recognition and analysis projects. Customers request for tools and cost-effective solutions in human supervised creation of truth data. A framework for developing these solutions is presented in the following.

The truth data model is presented in section II. The truth data capture process and toolkit architecture are outlined in section III. As an example, the use case of letter recognition in mail processing and the corresponding tool is shown in section IV, alongside a set of suggesters in section V. Implementation issues are discussed in section VI, followed by remarks on evaluation and results in section VII. Observations made in this project and an outlook to further work conclude this paper.

II. Truth Data

Truth data has general properties which are explicitly modeled in this work. Basic property of truth data is that it is pre-

978-1-7281-5054-3/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDARW.2019.10030

CPS Conference Publishing Services

path	class	content	transient	conf	status	creator	date
/distribution.1/code	Text	22788002	[]	1.0		suggest_distribution_code	2019-07...
/input.1/mailpiece_image	Image	file:///home/schamba/work/201...	[]	1.0		source	2019-07...
/receiver.1/elements.CNT/reference	DirElement	{}	{'alts': ['DEUTSCHLAND']}	1.0		suggest_syntax_elements	2019-07...
/receiver.1/elements.CTY/location	Textrange	{'ranges': [[11, 18]], 'refers': '/rec...	[]	1.0		suggest_syntax_elements	2019-07...
/receiver.1/elements.CTY/reference	DirElement	{'name': 'HAMBURG'}	{'alts': ['HAMBURG'], 'd...	1.0	confirmed:ignore	suggest_syntax_elements	2019-07...
/receiver.1/elements.DPC/reference	DirElement	{}	{'alts': ['22788002']}	1.0		suggest_syntax_elements	2019-07...
/receiver.1/elements.ORG/reference	DirElement	{}	{'alts': ['DEUTSCHE AN...	1.0		suggest_syntax_elements	2019-07...
/receiver.1/elements.PCA/location	Textrange	{'ranges': [[5, 10]], 'refers': '/rece...	[]	1.0		suggest_syntax_elements	2019-07...
/receiver.1/elements.PCA/reference	DirElement	{'name': '22788'}	{'alts': ['22788'], 'attribu...	1.0	confirmed	suggest_syntax_elements	2019-07...
/receiver.1/image	Image	file:///home/schamba/work/201...	[]	1.0	confirmed	edit_gen_region	2019-07...
/receiver.1/line.1/image	Image	file:///home/schamba/work/201...	[]	1.0		suggest_blocks_opencv	2019-07...
/receiver.1/line.1/location	Polygon	{'coords': [[0, 0], [0, 115], [277, 1...	[]	1.0		suggest_blocks_roi	2019-07...
/receiver.1/line.2/image	Image	file:///home/schamba/work/201...	[]	1.0		suggest_blocks_opencv	2019-07...
/receiver.1/line.2/location	Polygon	{'coords': [[31, 179], [31, 353], [8...	[]	1.0		suggest_blocks_roi	2019-07...
/receiver.1/location	Polygon	{'coords': [[1362, 1027], [1715, 10...	[]	1.0	confirmed	edit_gen_region	2019-07...
/receiver.1/records	Records		[{'CNT': ['DEUTSCHLAN...	1.0		suggest_od_records_ads	2019-07...
/receiver.1/script_type	Enum	HAND	[]	1.0	confirmed	suggest_read_text_decomp	2019-07...
/receiver.1/selections	Selection	[{'CNT': 'DEUTSCHLAND', 'status':...	[]	0.0		suggest_od_selections	2019-07...
/receiver.1/text_nom	Text	DAK 22788 Hamburg	[]	0.93...		suggest_read_text_decomp	2019-07...
/receiver.1/text_ref	Text	DAK 22788 HAMBURG	[]	1.0		suggest_normed_text	2019-07...
/receiver.1/tree	Tree		[{'alts': [], 'children': [{'a...	0.0		suggest_od_tree	2019-07...

Fig. 1. Truth data extracted from German letter mail document. Data semantics on the left side (element, spec), content in the middle (location, refers, content, alts), status on the right side (creator, date, confidence, status).

liminary and never complete. *Preliminary*, because correction of truth data is a common process during the development of recognition and analysis systems; *never complete*, because new requirements during system development – external or internal – create the need for new truth data elements. Considering this transient aspect of truth data, truth data is modeled by atomic items, with each having explicit references to related items, if needed.

Saying this, the general truth data model is simply a set of truth data items, each representing a single piece of information only. Each item consists of:

1) Content - a single piece of information
2) Semantics - a specific name for item's meaning
3) Status - info on truth data capture process

(1) The content is represented by specific classes. Common examples for classes are *Image*, *Polygon*, *Text*, *TextRange*, or *Enumeration*. The actual representation of the information is defined by the class, e.g. an image may be represented by a TIFF buffer. As the item's information is "atomic", even closely connected data is modeled by separate items: Image regions may be defined by a polygon, however the extracted image buffer is a different item, because image transformations may contain truth in its own right, e.g. noise removal. However, as atomic information nevertheless has to be complete, some truth items need additional references. This especially holds for locations: An image location represented by a polygon needs the reference to the image the polygon points to.

The truth data model is similar to [2] which is mainly focused on image data. Here, any kind of information may be stored, especially text and syntactic labels, which plays an important role in many projects. The model can be extended to multi-image, video, audio and other sensor data.

(2) The semantics of the item is given by a unique identifier. It consists of element names, e.g. "input", "receiver", "image", and an optional specifications, e.g. indices. By common prefixes, the identifiers are implicitly structured in a tree hierarchy. Examples of prototypical identifiers in mail processing applications are given in Fig. 2. The model is specified by a schema which defines valid identifiers, their classes and relationship. An example of truth data extracted from German letter mail can be seen in Fig. 1.

```
/input.1/mailpiece_image
/receiver.1/location
/receiver.1/image
/receiver.1/line.1/location
/receiver.1/line.1/image
/receiver.1/elements.cty/location
/receiver.1/elements.str/location
```

Fig. 2. Subset of truth data items identifiers to illustrate the data model structure. Truth data items are organized in a tree structure, which represent semantic dependencies: Changes in parent elements invalidate child elements. Element specifications introduce an additional hierarchical level, e.g. "line.*1*", "elements.*cty*". Other relations, e.g. references to images in locations, are part of the data content.

(3) Finally, the item's status is crucial for the truth data capturing process, which will be presented in the next section. In this process, the following status information is used:

- Creator, either a user name or a suggestion algorithm id.
- Date of last change.
- Confirmation status: *suggested–confirmed*.
- Control status: *ignore–use*

- Suggestion confidence.

Automatic suggestion is controlled by this status data. For example, truth data suggestion is switched off for confirmed data. In addition, truth data lifetime spans longer periods with potentially multiple truthing sessions, so transparency of the data capturing process is necessary. For reproducible usage, truth data then also has to be under version control.

Note the similarity of the truth data model to classical file systems: Identifiers correspond to file names, atomic information to file content, their classes to MIME types, and status information to file permissions. This makes the data model generic enough for many use cases, while preserving enough structure for steering the truth data capture process.

III. Truth Data Capture Process

The process of truth data capture ("truthing") is iterative on two levels. On data set level, documents are edited multiple times by various persons and algorithms, under various aspects, at different times. On document level, the truth items are worked on in any order, processing the document iteratively. This process of truthing a single document is sketched in Figure 3.

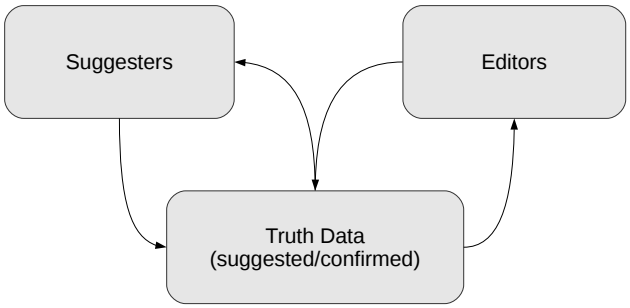

Fig. 3. Truthing process: Starting from initial *Truth Data*, *Suggesters* generate new truth data items. These are displayed to the operator by *Editors* and may be confirmed or modified, thus updating the truth data items. After each operation, suggesters are invoked again, until the operator decides to finish the process an store the data.

The truthing process is a loop which iterates between automatic suggestions and operator input, until the operator decides to finish the loop and store the data. Based on operator input, suggesters are able to provide better suggestions in each iteration. While editors are allowed to change any truth item, suggesters can only change those items which are not confirmed.

In order to additionally assist the human operator, suggesters provide confidences; these can be visualized in the editors in order to steer the editing process while preserving the operators' autonomy. In the case of overall high confidence, this even can be the recommendation to simply confirm all truth items. In the case of already confirmed items, it can indicate items that are confirmed but nevertheless wrong.

The process starts with either empty or existing truth data. One reason for starting with non-empty truth data is existing truth data from other sources which has to be revised. More often, existing truth data has to be enriched with additional truth data elements, or improved suggest algorithms are used to revise the truth data.

While the suggester algorithms seem to correspond to a standard recognition and analysis system, subtle differences should be noted. While those systems generally are tuned to generate minimal error at the cost of higher reject, suggesters can use brute-force strategies at minimal risk, because errors are corrected at low additional cost. This allows the cost- and time-effective usage of generic recognition systems which are not perfectly tuned to the respective task. Indeed, this even improves the utility of created data, because testing the shortcomings of one system is best done with truth data created by a another system, thus making virtue out of necessity.

For editing, there are basically two strategies: In case of a single fatal suggestion error, e.g. a missed text block, the first strategy is to correct this error, allowing the suggesters to provide correct dependent results in the next iteration. This works in cases with clear unambiguous sequence of computation. However, in many recognition tasks, there is a mutual dependence on data elements, in some context sometimes labeled "Sayre's Knot", e.g. character segmentation and recognition. But it holds for many other analysis tasks, e.g. text recognition and syntactic labelling, or syntactic labelling and semantic assignment. Here, the second strategy is most effective: Giving the decisive hint, thus improving the results in a "virtuous circle". This is the core improvement of the framework presented in this work.

Unconfirmed truth data, which only has been suggested, can be created by skipping the editor step, thus running suggesters in batch mode. This way, big amounts of data can be processed at limited cost. Regarding the increase of documents available, this aspect is important. The data is useful for many tasks: Filtered accordingly, it can be used for training of machine learning algorithms, or as basis for selecting those documents where manual truthing is worthwhile.

Regarding the outer document data set loop, several aspects have to be considered. Documents have to be selected and managed based on criteria from previous truthing steps, user management improves data quality by multiple assignment of tasks, and data version control has to be handled automatically.

IV. Truth Data Editors

While, strictly speaking, suggesters are optional in the presented work, an editor must exist for any truth element that has to be captured in the specific task. The reason for this is that any non-trivial suggester may fail for some documents. Conversely, any standard truth editing system can be enhanced by adding suggest algorithms.

The use case for the truth editors presented here is the annotation of letter mail images with the receiver address. The following truth data elements are captured:

- Receiver block location
- Receiver block image
- Receiver line locations
- Receiver line images

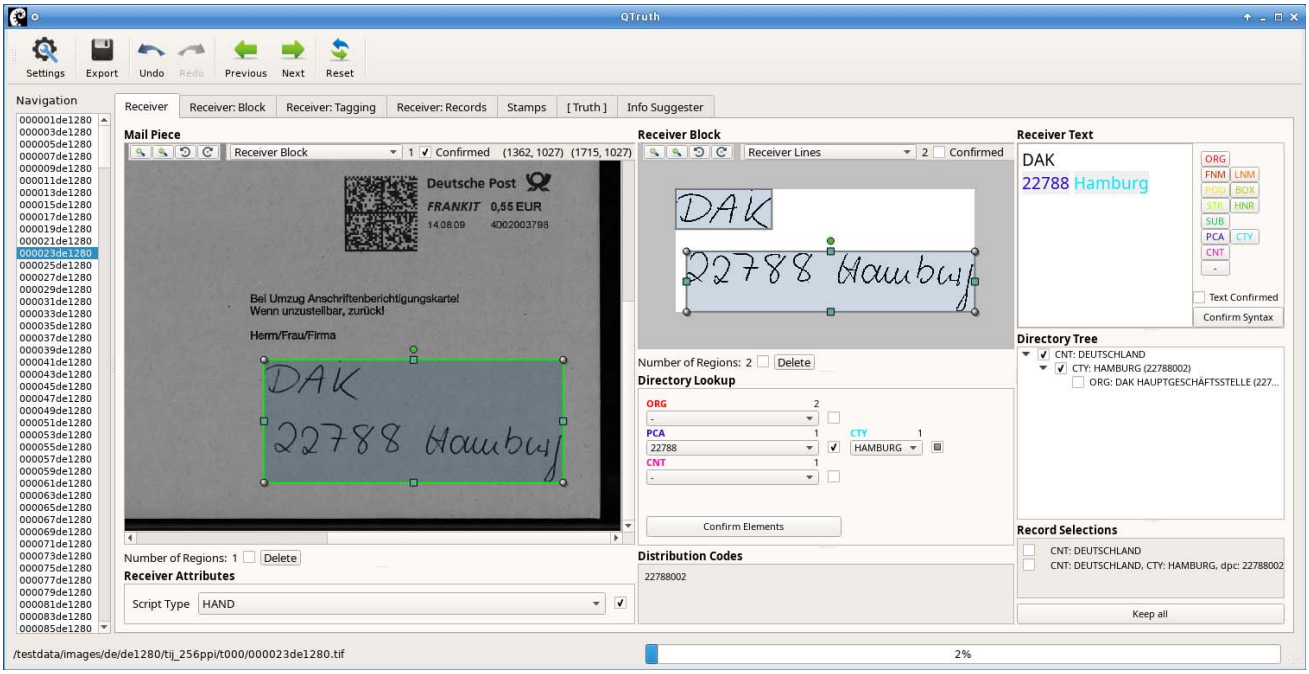

Fig. 4. Truth editors, applied to German mail recognition project: Left side: On top selection of receiver address location, on bottom suggestion for script type handwriting. Middle column: Line locations within the receiver address block, selection of address directory fields, and display of synthesized postal company distribution codes. Right side: Editing of text and syntax label assignment, selection tree for address database records, and display of selected partial records.

- Receiver text
- Receiver script type (handwriting/machine print)
- Receiver syntax tagging
- Receiver address records (database-specific)
- Receiver address fields
- Receiver distribution codes (customer-specific)
- Stamp locations
- Stamp images
- Barcode locations
- Barcode images

For each truth element, at least on editor must be provided. In some cases, e.g. the selection of address database records and fields, alternatives are provided for selection in the editor programmatically. In other cases, data is generated deterministically, e.g. images are cut by given polygons. In these cases, the task of the editor is to select and confirm only. Technically, the functionality is implemented as "suggester" even though being strictly deterministic.

A screenshot of the application editor is in Fig. 4.

V. Truth Data Suggesters

The suggestion step during truth processing as in Fig. 3 is performed by a set of atomic suggesters. Each atomic suggester creates or modifies a subset of truth data elements given another set of truth elements as input. Based on this input–output relations, truth data elements form a graph which may contain cycles. As discussed in section III, this appears to be beneficial for the truthing process. However, for computation, a sequence of computation has to be defined.

First there may be the notion of a "natural" sequence. Given a set of S of $|S|$ suggesters, a set T of $|T|$ truth data elements, and a subset T_0 of initial truth elements. For example, the initial element may be the document image. A "natural" suggester sequence $s_1 \dots s_{|S|}$ may be constructed by induction as follows: For each timestep $t = 1 \dots |S|$, select one suggester s_t from the remaining suggesters $S \setminus \{s_1 \dots s_{t-1}\}$, whose mandatory input is contained in T_{t-1}, creating a new set of truth items T_t. If multiple suggesters fulfill the condition, choose one of these randomly; if no suggester fulfills the condition, do so with all remaining suggesters.

Given this construction method, it is obvious that the sequence is not deterministic. When each suggester takes all truth data as input and changes it, the sequence is random. However in general, the presented sequence is reasonable: For a linear set of suggesters s_n, each constructing truth t_n from input t_{n-1}, the sequence is $s_1 \dots s_{|T|}$ as expected.

For truth calculation, the sequence of suggester calculations is determined in a similar way. Given existing truth data (initial, or from the last editing step), the next suggester s_t is chosen the following way: From the set of remaining suggesters $S \setminus \{s_1 \dots s_{t-1}\}$ choose one whose input truth elements are available and more recent than output elements. For efficiency, check this condition in the natural order of suggesters. Stop if no suggester fulfills the condition. If suggesters are cyclic, the whole procedure is repeated one

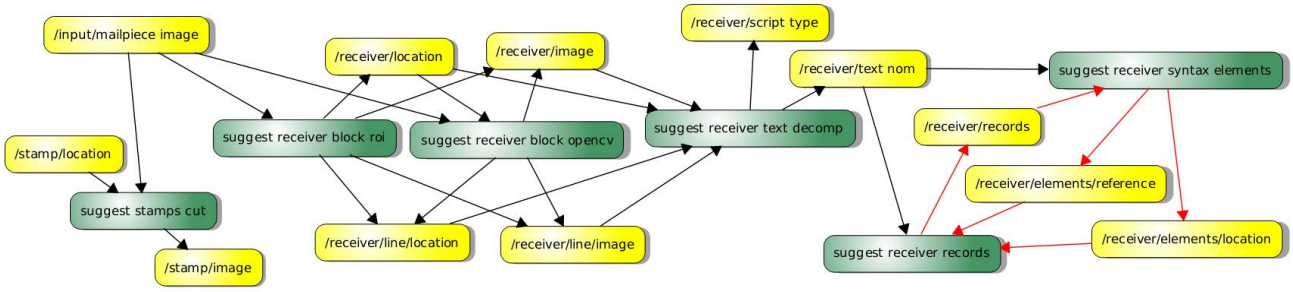

Fig. 5. Suggester structure: Yellow nodes are truth elements, green nodes are suggesters. Based on the mailpiece image (upper left), suggesters derive other truth items. Cyclic dependencies are shown in red. Note that some truth elements (stamp location, on the left) have no source, but must be manually edited, because no suggester exists (yet).

time.

Note the similarity to a *make* process in software builds. In contrary to these, dependencies of artifacts – here truth items – are cyclic. This poses no problem, because contrary to software builds, artifacts are allowed to be incomplete, and to iteratively influence and change each other. In interactive document analysis applications, cyclic dependencies of algorithms can be handled and are beneficial, because in many cases they well suit the problem domain. They can also be used for offline systems, as convergence properties play a minor role given the "suggestion" character of recognition and analysis tasks. Moreover, managing a set of possibly cyclic suggest algorithms facilitates the engineering of document analysis systems.

In this work, the following suggest algorithms have been used:

- Text block and text line detection based on similar methods as presented in [3].
- Image cutting and binarization. [4]
- Text recognition based on RNN/LSTM and CTC [5], and using Tesseract [6].
- Script type recognition.
- Error tolerant address database access with Solr [7].
- Selection of address fields based on text and address records.
- Syntactical tagging as presented in [8].
- Generation of mail company distribution codes.

Note that some dependencies of these suggesters are cyclic: Address database access operates on text, address fields and syntactical tagging, while syntactical tagging operates on text and address fields. The complete suggester structure is shown in Fig. 5. Further cyclic suggesters are envisaged: Text recognition additionally based on address field selection may improve recognition accuracy.

The choice of suggester implementation influences the quality and utility of truth data. Because in many cases, truth data is ambiguous by nature, e.g. handwriting, segmentation, interpretation; thus the result of truth data capturing is highly dependent on suggestions, if provided. This creates the risk of over-adaptation: Machine learning training may be less robust, test results may be less relevant. That's why it is useful to apply "orthogonal" algorithms, i.e. those producing results differently compared to the system to develop. Third-party and general purpose software is appropriate; as truth data capture mostly happens in early stages of system development, their use is first choice anyway. If data acquisition and system development happens in parallel, update of suggesters may be useful.

VI. Implementation

Requirements for fast and cheap data acquisition are to provide the data capturing process without installation. A web application implements data and user management and provides the truth editors as presented in section IV. Additionally, a desktop application based on the Qt toolkit has been provided for development and testing of suggesters. Suggesters are REST based, similar to the REST services presented in [9]. This facilitates integration of 3rd-party suggest software. Truth data is represented in JSON format and persisted using git and Cassandra database. For dataset batch processing, tools for suggestion, import, export and evaluation of multiple documents have been provided. Import and export are essential because legacy data and subsequent tools for training and test rely on proprietary data formats.

VII. Evaluation and Results

The evaluation of the truth data acquisition process itself is difficult, because the main aspect of truth data – correctness – cannot be determined due to the missing definition: Which data, acquired with different tools and algorithms, is the correct one? For this type of evaluation, there simply is no "gold standard". However, other aspects can be evaluated: Truth data acquisition costs, and suggestion accuracy.

Truth data acquisition costs can be measured using various scales: User interaction, e.g. quantified in key strokes or mouse clicks, or acquisition time, i.e. the average time users need to finish document truthing. In both cases, usability aspects and users' capabilities play a role, so experiments have to be conducted. Usability is influenced by data presentation, interaction options, and workflow. In an early version of the system [8], a speed-up of up to 75% compared to a base

system without automatic suggestions has been shown. For applications that allow the usage of data which has not been manually confirmed, e.g. in machine learning, a speed-up factor of $10 - 20$ seems reasonable. This is based on the assumption that no suggesters for the relevant data existed before.

For completeness, the theoretical minimum number of user interactions is defined for a any set of data, suggesters, and editors, and could be evaluated. However, no attempts have been made so far to perform this kind of evaluation.

Suggester quality has been measured for text recognition and address database access. "Correct" truth data had previously been captured manually. It has to be noted that quality of this "correct" truth data is not perfect, so there is room for interpretation of results.

- For evaluation of text recognition, the average character error rate has been calculated using Levenshtein distance, ignoring case, but considering whitespace and special characters. On a set of 20k German letter mail documents, the receiver text has been suggested based on receiver block location, image binarization, line finding and text recognition. In comparison to manually typed text, 14.7% character error has been measured. Closer inspection of difference gave the following findings: In manually transcribed text, German special characters have been wrongly labelled, and optional lines (concerning address syntax) have often been skipped, which accounts for most errors, while in suggested text, whitespace and punctuation are the main causes of errors.
- For evaluation of address database access, four suggester methods have been compared during development. All methods are based on Solr/Lucene [7] and represent steps during development. A set of 100 German letter mail documents has been manually labelled with database records and corresponding tokens appearing in the address text. Results are given in Table I. The percentage of documents with correctly identified address tokens has been counted. During development, accuracy has been improved heavily. Closer inspection of manually labelled data did not show errors there.

TABLE I
EVALUATION OF ADDRESS DATABASE ACCESS: PERCENTAGE OF DOCUMENTS WITH CORRECTLY IDENTIFIED TOKENS.

	City	Street	House Nr	All tokens
# documents with token	93	79	79	100
Method 1	83%	33%	84%	28%
Method 2	97%	49%	**96%**	38%
Method 3	89%	76%	82%	59%
Method 4	**99%**	**95%**	95%	**76%**

During truth data capture and suggester evaluation, deficiencies of existing truth data are detected. This is a general concept while developing document analysis systems: Truth data is never complete. However, for future evaluation, careful tracking of truth data changes, using various suggesters and manual editing, could give deeper insight into the truthing process.

VIII. CONCLUSIONS

The usage of appropriate suggesters reduces the costs of ground truth data capture considerably. By allowing suggesters to introduce circular dependencies on the truth data they operate, the addition of new suggesters is easy. An algorithm has been presented, which automatically integrates any suggester working on the truth data. This way, sophisticated round-trip reasoning is rendered possible. Cyclic suggest algorithms show to be the key issue in speeding up truth data capturing, and may resolve Sayre's knot by imitation of human reasoning. Integrating the suggesters into the presented semi-automatic, interactive truthing process reduces the risk of a negative feedback cycle. Instead, correctly chosen user input reduces necessary interactions to a minimum.

REFERENCES

[1] M.-P. Schambach and S. von der Nüll, "Benutzerunterstütztes Automatisches Informationserkennungsverfahren bei Logistikprozessen," Patent Request 18 196 445.3-1217, Sep. 25, 2018.

[2] S. Pletschacher and A. Antonacopoulos, "The PAGE (Page Analysis and Ground-truth Elements) Format Framework," in *20th International Conference on Pattern Recognition (ICPR)*. IEEE, 2010, pp. 257–260.

[3] Z. Li, M. Schulte-Austum, and M. Neschen, "Fast Logo Detection and Recognition in Document Images," *2010 20th International Conference on Pattern Recognition*, pp. 2716–2719, 2010.

[4] N. Otsu, "A threshold selection method from gray level histograms," *IEEE Trans. Systems, Man and Cybernetics*, vol. 9, pp. 62–66, Mar. 1979, minimize inter class variance.

[5] A. Graves and J. Schmidhuber, "Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks," in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. Curran Associates, Inc., 2009, pp. 545–552.

[6] R. Smith, "An overview of the tesseract ocr engine," in *Proceedings of the Ninth International Conference on Document Analysis and Recognition - Volume 02*, ser. ICDAR '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 629–633. [Online]. Available: http://dl.acm.org/citation.cfm?id=1304596.1304846

[7] S. Handiekar and A. Johri, *Apache Solr for Indexing Data*. Packt Publishing, 2015.

[8] S. Gupta, "Approximate Search in Address Directories for Efficient Ground Truth Data Creation," Master's thesis, Technical University of Chemnitz, 2017.

[9] M. Würsch, R. Ingold, and M. Liwicki, "SDK Reinvented: Document Image Analysis Methods as RESTful Web Services," in *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*, vol. 00, April 2016, pp. 90–95. [Online]. Available: doi.ieeecomputersociety.org/10.1109/DAS.2016.56