ASLForm: An Adaptive Self Learning Medical Form Generating System
Shuai Zheng, MS1, Fusheng Wang, PhD2,3, James J. Lu, PhD1
1 Department of Mathematics and Computer Science, Emory University, Atlanta, GA
2 Department of Biomedical Informatics, Emory University, Atlanta, GA
3 Center for Comprehensive Informatics, Emory University, Atlanta, GA

Abstract
To facilitate the process of extracting information from narrative medical reports and transforming extracted data into standardized structured forms, we present an interactive, incrementally learning based information extraction system - ASLForm. ASLForm provides users a convenient interface that can be used as a simple data extraction and data entry system. It is unique, however, in its ability to transparently analyze and quickly learn, from users' interactions with a small number of reports, the desired values for the data fields. Additional user feedback (through acceptance decision or edits on the generated values) can incrementally refine the decision model in real-time, which further reduces users' interaction effort thereafter. The system eventually achieves high accuracy on data extraction with minimal effort from users. ASLForm requires no special configuration or training sets, and is not constrained to specific domains, thus it is easy to use and highly portable. Our experiments demonstrate the effectiveness of the system.

Introduction
Compared to traditional free-form text reporting system, "structured reporting" [1] offers significant promise for both human consumption and machine processing. Report standards, controlled vocabularies and terminologies have proliferated in medical domains to standardize the creation of medical reports. DICOM structured reporting standardizes reporting in radiology images [2]. Structured Reporting for Anatomic Pathology [3] is under development by IHE for standardized structured pathology reporting. Such structured reporting often depends on pre-defined templates or vocabularies. Examples include the College of American Pathologist's (CAP) cancer protocols and checklists [4], which provide detailed checklists for options in pathology reports. The Cancer Biomedical Information Grid (caBIG) [5]'s Cancer Data Standards Registry and Repository (caDSR) [6] offers APIs and tools to define common data elements (CDEs). HL7 Clinical Document Architecture [7] defines standardized document structure and semantics to share electronic health information. These efforts to standardize medical reporting formats and vocabularies give rise to the possibility for automated searching, browsing, and mining of medical data.

Meanwhile, most existing medical report systems are based on natural language narrations written in free-form text. While some high-level structures exist --- for example, patient records may contain sections on "Medications on Admission", "Hospital Course" and "Condition on Discharge" --- the narrative style of each section is often highly informal and personal. For instance, each of the following patient record snippets describes similar "heart rate" information of patients during physical examination:

1. "...... Blood pressure was 152/63 , heart rate 67 with occasional premature ventricular contractions, respirations 15 ......"
2. "......Her pulse was regular at 82 beats per minute......"
3. "......The blood pressure was 115/73 and heart rate was 93......"
4. "......122/66 , 96.8 , 81 , 21 for vital signs......"

Large amount of unstructured text based information is difficult for humans to browse efficiently and effectively. It is even more challenging for machine processing. To utilize such reports, valuable information needs to be extracted either manually or with automated tools. For large datasets, manual extraction is tedious and time-consuming. Software tools, on the other hand, are typically tuned for particular domains, and precisely annotated
training datasets need to be developed and learned to establish decision models through which subsequent reports can be processed. Once trained, the decision model is difficult or impossible to modify and improve. In addition, automatically extracted data suffers from inaccuracies as a result of natural limits on statistical machine learning techniques.

For biomedical research that requires completely accurate data, the above discussion points to one constant: human involvement is necessary, whether to perform the actual extraction or for post-extraction verification. The goal of ASLForm meets these requirements in a more convenient and intelligent way. Using information gathered from ordinary user interaction during manual text extraction, ASLForm quickly learns, proposes and generates answers for pre-defined form data elements automatically. Moreover, from user feedbacks, the system continues to learn and improve its ability to find answers, in real-time, without interruption or the need for special processing. Besides, no expensive initial training data is required. The self-adapting, self-improving decision model also translates to diminishing efforts from the user. As the collection of reports is processed, the user’s role evolves from extracting/populating information to verifying system-proposed information. ASLForm is generic and not limited to a specific domain or text structure. Users can also customize output forms and formats or supply standardized report forms. The GUI interface is intuitive and convenient to use. It extracts answers for single-valued attributes, such as age and gender, which can take one value, as well as multiple-valued attributes such as history of diseases and prescriptions that may take multiple values. Our initial performance study demonstrates high effectiveness of the system, and we are currently adopting the system for a large scale cohort identification project at Emory University.

Background

A number of research efforts have been conducted in the field of medical information extraction. cTAKES (clinical Text Analysis and Knowledge Extraction System) [8] is an open-source NLP system designed for extracting information from clinical text. It offers various NLP tools trained especially for clinical fields. Most algorithms or systems focus on a particular application domain such as pathology reports [9, 10] or biomedical text [11, 12]. caTIES (Cancer Text Information Extraction System) [13, 14] is a cancer text information extraction system specialized in tissue annotations. ONYX [15] adapts semantically annotated grammar rules to analyze sentence level text. MedLEE [16, 17] (Medical Language Extraction and Encoding system) is a clinical information extraction system that offers the feature of mapping information to controlled vocabularies.

Many applications emphasize the value of special information within text. NegEx [18] specializes in identifying negated or possible trigger terms in clinical text, as well as the affected scope of the term. TEXT2TABLE [19] values the importance of medical events, which is based on modality events, time and negative information.

Adaptive machine learning provides customizations for different working environments and supports incremental improvement. Amilcare [20] is an adaptive information extraction system used for Semantic Web annotation. Its algorithm, (LP)3 [21], generalizes and induces symbolic rules. DUALIST [22] allows users to select system populated rules for feature annotation to support text classification, word sense disambiguation and information extraction. Another related research area is interactive annotation, which attempts to ease the annotation process by incorporating machine learning techniques. MIST [23]’s classifier learns automatically to support de-identification. RapTAT [24] learns document phrases to accelerate annotation.

Recent work also includes intelligent systems that can comprehend medical information and deduce conclusion. Examples are i2b2’s [25] Obesity Challenge [26] of inferring "who's obese and what co-morbidities do they (definitely/likely) have" and the challenge of identifying the smoking status for patients [27].

Methods

Overview

ASLForm’s interface and workflow conform to regular reporting systems: a user browses the input text and fills out an output form. The system empowers this standard process with three special features: i) The system fills the output form automatically upon loading each text input; ii) The system adapts its information extraction model
automatically from user feedbacks during the standard process; and iii) The system does not need predefined constraints, nor does it require pre-training with labeled data.

Extracting information from the input text to fill a structured form is performed in two steps. First, regions where target values may appear are detected by a combination of locations in the text and co-occurring words. In this process, constraints such as regular expressions and ranges of numerical values may be applied to narrow the search for candidate values. Secondly, features such as part-of-speech tags and adjacent texts are used to classify candidate values. Candidates that receive confidence scores above the threshold are filled into the output form, which could be later transformed into a standardized format based on controlled vocabularies.

The information extraction model of ASLForm is adapted gradually as the collection of text input is processed. This is realized by the statistics of features and an updatable neural network. As a user reviews system-generated values, unrevised values and manually updated values are treated as correct answers. Features associated with these "ground truths", such as whether a value is a noun or a verb, a numeric or a string, are exploited to update the decision model. In general, the system becomes more intelligent and mature with increasing number of samples.

Features of this system are generic and not limited to specific domains, thereby making the system more portable to different environments. The workflow of the system is consistent with most existing reporting systems, where users can read input text and fill an output form. This provides the possibility of integrating ASLForm easily with different applications.

Figure 1. An example screenshot of ASLForm’s interface

Interface

As shown in Figure 1, the main window of ASLForm’s interface is split into two main panels that sit side by side: a text input panel and an output panel. The text input panel displays the current document being processed, and the output panel displays extracted name-value pairs of data elements of interest. The output panel contains a three-column table. The "index" column uses colors to highlight locations of values of the form data elements in the text input panel. The "Attribute" and "Value" columns show the attribute names and associated values of form data elements respectively. The "Previous" and "Next" buttons at the bottom on the right-hand side allow users to navigate through the input document collection. A user can review and update automatically populated values either through mouse-highlighting the value in the text followed by a click on the data field, or through manually typing in. The interface is simple and intuitive, and the underlying text processing and learning process is transparent to users.
**Workflow**

![Workflow Diagram](image)

**Figure 2.** The workflow of the system

The workflow of ASLForm is shown in Figure 2. A user specifies an input document collection and immediately begins to process them manually. Upon loading each document, the "Output Form" is filled automatically, as completely as possible, with extracted values. Locations of these values are highlighted in the text input panel. The user reviews system generated values, revises and updates them if necessary, before moving to the next document. In between documents, the system learns features of correct and incorrect answers by comparing generated values and manually revised ones, and updates the decision model accordingly. Through this revision process, the amount of information that the system is able to correctly prefill grows over time; the "Output Form" for the first few documents may be empty.

If the user is sufficiently confident about the system's decision model, the user has the option to turn off the review to allow the system to run in batch mode. ASLForm will process the remaining documents directly without reviews. The interface and workflow can be found in a demo video in the wiki [28].

**Architecture Overview**

The system consists of the following major components: preprocessing, answer generating, and adaptive learning. Figure 3 shows relationships of the components with respect to how the data flows. Blue boxes indicate results or inputs to components and subcomponents. The preprocessing component converts input texts and output forms into internal data structures used by the answer generating component. The answer generating component extracts values from input texts to fill the output forms. The adaptive learning component utilizes judgments of users, given in the form of edits on generated values, to update the decision model of the answer generating component. Next we introduce the modules of each system component.

![Architecture Diagram](image)

**Figure 3.** System components and dataflow
**Preprocessing Component**

The preprocessing component includes Text Parser, Indexer and Form Builder, as described below.

*Text Parser.* The Text Parser detects the underlying structure of plain text and analyzes nature language features. ASLForm relies on Apache OpenNLP [29] for text processing. Upon loading a document, the text content of a report is parsed into a hierarchy consisting of four layers: section, paragraph, sentence and token. Advanced nature language features such as part of speech (POS) and named entity are also analyzed.

*Indexer.* The Indexer creates an index for tokens in text from which the location of a given string can be obtained. During searching, other natural language features are returned along with location information. For example, given a string, the Indexer can return the section, paragraph, sentence and phrase the string belongs to, as well as part of speech and named entity information of the string.

*Form Builder.* The Form Builder provides a wizard with which metadata of the output form can be constructed. The user builds a form by specifying a list of form data elements and defining features for each element. Except for the names of the form data elements, no other features require predefinition; they all can be learned gradually (see the "Adaptive Learning Component" section for more details). The user also has the option to define constraints manually to improve the decision model. Examples include specifying the sections that should be targeted or the range of numerical answers such as heart rate, temperature and blood pressure.

**Answer Generating Component**

The answer generator produces answers for form data elements with the support of three sub systems: querying module, verifying module and standardizing module, as shown in Figure 4.

![Figure 4. Modules of the Answer Generator](image)

**Querying Module**

The querying module generates a list of candidate values in three steps, described as follows.
First, it locates sentences of possible values using the following search strategies. i) Hierarchical searcher: Based on the text hierarchy generated by the text processor, the searcher identifies positions where target sentences are likely to exist. For example, in a pathology report, the name of the pathologist occurs in the first sentence of "Comments" section with high probability. ii) Landmark searcher: This locates target sentences by words that frequently co-occur with answers. For example, the target sentence of "specimens' received status" may always contain words "specimens" and "received". iii) Dictionary searcher: This finds target sentences using previously found answers. For example, if a sentence contains already collected disease names such as "myocardial infarction" and "diabetes mellitus", it is a likely target sentence for "familial diseases".

It then explores potential phrases from located target sentences through two chunkers, boundary model chunker and dictionary chunker. The boundary model chunker adapts the boundary classification information extraction model of BWI [30]. By detecting beginning and ending boundaries, possible phrases can be identified. The dictionary chunker uses accumulated dictionary to detect phrases. It seeks the sequence of tokens that has the longest match from target sentences.

Last, it eliminates incorrect candidates by applying appropriate filters. i) Part of speech (POS) filter: This filters a phrase by its POS tag in the sentence. Simple example phrases are nouns and verb phrase. ii) Named entity filter: This approves or rejects a certain type of named entity, and the current implementation supports basic named entities such as title and time. iii) Regular expression filter: This looks for chunks that match predefined string patterns. For example, the size of specimen always follows the pattern: *.* X *.* X *.* cm. iv) Value filter: This eliminates numerical or enumerated values that fall outside the specified range of values. v) Negation filter: This removes phrases governed by negation that reverses the meaning of the phrase. For example, for the text "the patient's parent has no history of cancer", "cancer" should not be extracted as a family disease.

After filtering, candidate values are extracted from target sentences, and results are forwarded to the verifying module for further analysis.

(II) Verifying Module

The verifying module examines candidate values in contexts. For a single-valued form data element, the value with the highest confidence score is used to populate the form. For a multi-valued form data element, for example, medication, values with confidence scores above an acceptance threshold are extracted. The module is implemented with a neural network classifier. Neural networks are naturally adaptive and able to approximate complex models in noisy environments. For each candidate value, the classifier considers its parts-of-speech tag, string value, and left and right neighbors. For the latter, words are considered if they appear before and after the value within a window of a certain size. For instance, to token "56" in the sentence "The patient is a 56 years old female", two-word and one-word grams before and after are used as part of the inputs to the classifier.

The classifier categorizes each candidate value as "accept" or "reject", with an associated confidence score. A popularity bias is then applied. That is, values that have been frequently selected previously are preferred over rarely selected values. If needed, these values are normalized by the standardizing module.

(III) Standardizing Module

The module normalizes values extracted from the input text to a form that is compatible with the user-defined format or a standard vocabulary. For example, for gender information, "gentleman" or "lady" may be mapped to "male" or "female". We are planning to improve this module through UMLS.

Adaptive Learning Component

The learning module collects information from a user's implicit feedbacks, consisting of the set of revisions to the values proposed by the answer generator. Two scenarios are considered for a given form data element in the target form. In the first scenario, the user advances to the next report without modifying automatically populated values. This indicates that the answers proposed by the system are correct. Therefore, features associated with the answers are learned as positive instances. In the second scenario, the user edits a value before proceeding to the next report. In this case, each system-generated answer that has been edited is treated as a negative instance since it has been rejected. Values filled in manually by the user are studied as positive instances.
Repeating the above steps, the system builds the answer generator model incrementally, with positive and negative instances collected gradually. Important features that the learning system collects are summarized in Table 1.

Table 1. Answer features

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Feature Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position</td>
<td>location of the answer in the text hierarchy</td>
</tr>
<tr>
<td>Landmark</td>
<td>words co-occurring with the answer in a sentence</td>
</tr>
<tr>
<td>Named entity</td>
<td>the type of named entity, or not a named entity</td>
</tr>
<tr>
<td>Regular expression</td>
<td>whether the answer matches a user defined regular expression or not</td>
</tr>
<tr>
<td>POS</td>
<td>parts of speech tag</td>
</tr>
<tr>
<td>Value</td>
<td>the string tokens of the answer</td>
</tr>
<tr>
<td>Neighborhood</td>
<td>adjacent words within a fixed window</td>
</tr>
<tr>
<td>Literal</td>
<td>Literal features such as capital, initial and special punctuation</td>
</tr>
</tbody>
</table>

Features of each value are inputs to either the querying module or the verifying module. Searchers, chunkers and filters in the querying module monitor the distribution of instances of each feature. Features with high significance, as determined by predefined threshold, are used to support the searching and filtering processes. For example, if a word w co-occurs with the target value in more than half of the instances, w will be used by the landmark searcher.

The classifier of the verifying module is adapted each time a record is processed, and the new model is applied to extract answers in (as well as further updated by) subsequent documents.

Results

Experiment Setup

While the system gives users the facility to predefine constraints, here we focus on examining the system’s ability to learn without predefined constraints or training data. In all test cases, the system starts with an empty model without prior training. To explore the accuracy and the rate at which the decision model improves, we perform several evaluations of the system on real world datasets from i2b2’s NLP Data. The i2b2's NLP Data contains several sets. Set #1C [27] contains 398 de-identified discharge summaries. Set #2 [31] was used in the 2008 Obesity Challenge, and contains 611 patient record. The records in these two sets contain dissimilar structures. For example, in Set #1C, 234 of the 398 records have the section "HISTORY OF PRESENT ILLNESS". To avoid performing experiments over records with missing information, we pre-processed and normalized four subsets of the data, summarized in Table 2.

Table 2. Testing dataset

<table>
<thead>
<tr>
<th>Subset Index</th>
<th>Original Dataset</th>
<th>Contained Section</th>
<th>Count of Reports</th>
</tr>
</thead>
<tbody>
<tr>
<td>subset 1</td>
<td>i2b2's NLP Data Set #1C</td>
<td>Contains &quot;History of Present Illness&quot; section</td>
<td>234</td>
</tr>
<tr>
<td>subset 2</td>
<td>i2b2's NLP Data Set #1C</td>
<td>Contains &quot;Physical Examination&quot; section</td>
<td>226</td>
</tr>
<tr>
<td>subset 3</td>
<td>i2b2's NLP Data Set #2</td>
<td>Contains &quot;Family History&quot; section</td>
<td>140</td>
</tr>
<tr>
<td>subset 4</td>
<td>i2b2's NLP Data Set #1C</td>
<td>Contains &quot;Discharge Medications&quot; section</td>
<td>124</td>
</tr>
</tbody>
</table>

We consider two types of extraction: single value extraction such as age or gender, and multiple value extraction such as family diseases or discharge medications.

Single Answer Extraction

In this test, we conduct 4 test cases to extract age, gender, heart rate and temperature. Table 3 shows the results of the four test cases. If the information to be extracted does not exist in the input, an "N/A" value will be recorded. If
the system does not propose "N/A" in the absence of relevant information, we consider any proposed answer as incorrect and records it as such.

**Table 3. Result of singular answers test cases**

<table>
<thead>
<tr>
<th>Singular Answer Test Cases</th>
<th>Dataset</th>
<th>Count of Reports</th>
<th>Count of &quot;N/A&quot;</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract patient’s age</td>
<td>subset 1</td>
<td>234</td>
<td>4</td>
<td>94.01%</td>
</tr>
<tr>
<td>Extract patient's gender</td>
<td>subset 1</td>
<td>234</td>
<td>15</td>
<td>91.45%</td>
</tr>
<tr>
<td>Extract patient's heart rate in physical examination</td>
<td>subset 2</td>
<td>226</td>
<td>101</td>
<td>94.24%</td>
</tr>
<tr>
<td>Extract patient's temperature in physical examination</td>
<td>subset 2</td>
<td>226</td>
<td>153</td>
<td>93.36%</td>
</tr>
</tbody>
</table>

**Figure 5. Precision rate changes over processed records**

As records are processed, the system captures important features of target values, and utilizes them to improve the effectiveness and accuracy of future extractions. The learning occurs quickly. For instance, when processing heart rate information, within a few instances, "heart" and "rate" are captured as keywords to support Landmark Searcher and the target value is recognized as an integer value. As new instances arrive, new keywords such as "pulse", "beat" and "rhythm" are learned. The precision values in Table 3 consider results throughout the whole process, including the initial learning stage. Results in Table 3 show high accuracy considering that all four test cases start without prior training or predefined constraints.

Figure 5 shows results for four attributes: age, gender, heart rate, and temperature. Records are divided into five groups, given in the order in which documents are processed. Precision rates for every 20% of the records are plotted. Comparing the first quintile to the last, we observe notable improvements. In particular, precision rates of age, gender and heart rate are increased from 85.10%, 74.46% and 89.13% to 100%, 100% and 93.33% respectively, with the largest increase occurring between the first and the second groups. The plot of temperature shows similar trend over the first four groups, but the precision of the last group drops to 86.66% due to high noise in the data. These trends show the viability of incremental improvements to the system's decision model based on information collected during manual extraction. Within a few iterations, ASLForm can begin to generate answers with quickly increasing accuracies.

**Multiple Answers Extraction**

Multiple-value extraction is a more challenging task. We conduct two test cases to extract discharge medications and family diseases. If a value for an attribute occurs multiple times, only the first occurrence is kept. Since each record involves an unspecified number of target values, we use precision, recall, and the F1 score to estimate the effectiveness of extraction over the entire process. Similar to the single answer extraction, we begin without any constraint or training data. Table 4 shows results.
Table 4. Results of multiple answers test cases

<table>
<thead>
<tr>
<th>Multiple Answers Test Cases</th>
<th>Dataset</th>
<th>Count of Records</th>
<th>Count of Values</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract family diseases from &quot;Family History&quot;</td>
<td>subset 3</td>
<td>140</td>
<td>187</td>
<td>89.44%</td>
<td>86.09%</td>
<td>87.73%</td>
</tr>
<tr>
<td>Extract medications from &quot;Discharge Medications&quot;</td>
<td>subset 4</td>
<td>124</td>
<td>561</td>
<td>81.13%</td>
<td>84.31%</td>
<td>82.69%</td>
</tr>
</tbody>
</table>

The first test case extracts family diseases appearing in the "Family History" section. There are 187 total target values, which include diseases of parents, siblings and other relatives. For this extraction, negation detection is significant. Examples of negation include "she denies familial diabetes" and "no history of cancer". The negation filter played an important role in the experiments, rendering an overall information extraction F1 score of 87.73%.

The second test case extracts medication names from the "Discharge Medications" section. The 124 patient records contain 561 target values. Compared to family diseases, the narratives for prescriptions are in a more free form. Even in such a noisy environment, the extraction could still achieve a F1 score of 82.69%.

Discussion

Ongoing work includes introducing medical report checklists and vocabularies into ASLForm. By using a controlled vocabulary as a predefined answer dictionary, the system can extract answers based on background knowledge. Such knowledge can also be used to enhance the Normalizer of the system.

We are also working with Emory Cardiology to make the process of chart review to identify study subjects more automated. This is achieved by extending ASLForm to support data extraction from multiple reports. This use case reveals the practical need to extract information from multiple data sources to generate an integrated output. For a patient, besides structured records, there may exist various medical reports such as history and physical reports, discharge summary, clinic notes, cath and lab reports. Data fusion through combining data elements extracted from these reports and structured records provides comprehensive knowledge for much more accurate patient identification. This scenario also raises interesting research questions. For example, how should conflicts between different data sources be resolved, and how are conclusions drawn when multiple facts are present?

To maximize its applicability and portability, ASLForm currently employs generic tactics to support information extraction. However, we also recognize that given a particular task, domain specific information extraction strategies are likely to be highly effective. Future designs will allow users to empower the system with user-defined plug-ins such as new filters. This will enable the system to be specialized for data extraction problems with eccentric textual patterns.

Conclusions

While considerable attention has been given to structured and standardized reporting, most medical reporting systems still allow for (and thus encourage) narrative text descriptions. There is a lack of effective tools to ease the process of information extraction, data transformation and normalization. ASLForm provides a bridge between free-form text reports and structured reports. Its work flow follows the conventional process for manual extraction of information from text, but it noninvasively learns, and gradually improves its ability to automatically locate relevant information. As companion features, ASLForm also provides a solution for normalizing and standardizing the extracted data according to controlled vocabularies and medical report standards. The usability and adaptability to different working environments makes ASLForm a powerful tool. ASLForm will be packaged as a java library and made publicly available through its wiki [28].

References