Applied Medical Code Mapping with Character-based Deep Learning Models and Word-based Logic

John T. Langton
Wolters Kluwer Health
230 3rd Avenue
Waltham, MA 02451
jlangton@wolterskluwer.com

Krishna Srihasam
Wolters Kluwer Health
230 3rd Avenue
Waltham, MA 02451
ksrihasam@wolterskluwer.com

Abstract

Logical Observation Identifiers Names and Codes (LOINC) is a standard set of codes that enable clinicians to communicate about medical tests. Laboratories depend on LOINC to identify what tests a doctor orders for a patient. However, clinicians often use site-specific, custom codes in their medical records systems that can include shorthand, spelling mistakes, and invented acronyms. Software solutions must map from these custom codes to the LOINC standard to support data interoperability. A key challenge is that LOINC is comprised of six elements. Mapping requires not only extracting those elements, but also combining them according to LOINC logic. We found that character-based deep learning excels at extracting LOINC elements while logic-based methods are more effective for combining those elements into complete LOINC values. In this paper, we present an ensemble of machine learning and logic that is currently used in several medical facilities to map from custom codes to standard LOINC values.

1 Introduction

LOINC supports several use cases in the medical domain. For instance, a doctor can use LOINC codes to precisely indicate which blood tests they want a laboratory to perform. A major challenge is that clinicians often use custom codes when entering data into medical records systems. Custom codes may be more intuitive for humans to understand but also suffer from personal nuance and error. They can contain misspellings, shorthand, and invented acronyms. Further, custom codes are site-specific such that the codes in one facility may differ from those in another. These differences make it difficult for facilities to communicate. For instance, if a doctor uses one set of codes to order tests, and a laboratory uses a different set of codes to perform tests, then the laboratory may not be able to correctly identify which tests to perform what tests the doctor is ordering. It is necessary for software solutions to map from custom codes to standards like LOINC to eliminate differences between the codes that facilities use and support data interoperability.

2 The Task

There are around 40,000 LOINC codes. Each code contains six elements as shown in Table 1. Our task is to first extract the six elements from a noisy input string (e.g. custom code), then combine those elements to form a standard LOINC output. Equation (1) shows a real-world example with a custom hospital code on the left mapped to a standard LOINC code on the right. The following are five additional input strings from different hospitals that refer to the exact same LOINC code: {*"Ur Leukocyte Esterase", "LEUK ESTER", "UR Leuko Est–Clinitek", "LEUKOCYTE E URINE", "Leuk Est Test Strip U"*}. One can readily see the differences that complicate communication and data interoperability.

$$\text{U Leuk Est} \rightarrow \begin{cases} \text{5799-2 Leukocyte esterase,} \\ \text{Urine, Ordinal, PT,} \\ \text{Test Strip, Presence} \end{cases} \tag{1}$$

3 Data

The data for our project came from prior mappings that were performed manually by clinical informaticists. The distribution of LOINC values was skewed, with ten codes making up 87.7% of the data. The remaining codes made up a long tail distribution but not all possible LOINC codes were present. Table 3 shows the possible unique LOINC element values along with the coverage of those values in the available data. If we treat each LOINC

Proceedings of the 1st and 2nd Workshops on Natural Logic Meets Machine Learning (NALOMA), pages 7–11
June 16, 2021. ©2021 Association for Computational Linguistics

Element	Example	Description
Component	Leukocyte Esterase	What is being measured, observed, or evaluated
Specimen	Urine	specimen type collected for measurement
Scale	Ordinal	the scale of measure such as ordinal or nominal
Timing	PT	interval of time for measurement or observation
Method	Test Strip	Method of measurement
Property	Presence	Property of what is being measured such as mass or volume

Table 1: The six LOINC elements with examples.

Challenge	Example
Invented acronyms and missing letters	Lkct for Leukocyte
Misspellings	Luykocite
Missing delimiters	UrLeukEst
Missing LOINC elements	Leuk Est (component only)
Parsing and input / output errors	00001

Table 2: Challenges of mapping custom codes to LOINC standards.

code as a class, then the data distribution corresponds with severe class imbalance.

Medical facilities rarely have local codes for every possible LOINC. Instead, they maintain a subset of codes that are most commonly used in their practice. As a result, custom codes at a facility often exclude LOINC elements that are irrelevant to their practice. For example, a blood laboratory may exclude specimen because they implicitly know the value is always "blood". Our data, therefore, contained many codes with only a subset of the elements necessary to specify a full LOINC. Efforts to map custom codes to LOINC standards must contend with several challenges as enumerated in Table 2.

4 Related Research

Both machine learning and logic-based methods for NLP struggle with noisy text inputs. Vectorization methods including *Term Frequency – Inverse Document Frequency* (TF-IDF) vectorization (Xu et al., 2009) and word embeddings (Kim, 2014; Pennington et al., 2014) are particularly sensitive, though the use of sub-words (i.e. n-grams of characters) can somewhat ameliorate the issue (Edizel et al., 2019). Pre-trained transformer models have recently topped some NLP benchmarks but are also sensitive to noisy text (Devlin et al., 2018; Wang et al., 2019; Rajpurkar et al., 2018). Pruthi, Dhingra, and Lipton have shown that misspellings reduce BERT performance by significant margins and propose an independent model for spelling correction (Pruthi et al., 2019). Luong and Manning have used hybrids of character and word based recurrent neural networks (RNN) to address unknown words in translation (Luong and Manning, 2016). Zhang and Yang have used a lattice of Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) models (variants of RNNs) at the character and word level to improve performance of named entity extraction (Zhang and Yang, 2018). We evaluated multiple methods and determined to use a similar hybrid approach for extracting canonical LOINC terms from noisy text inputs.

5 Hybrid Solution for LOINC Mapping

We evaluated several methods for addressing noisy text inputs. Logic-based methods were paired with fuzzy matching, word frequency analysis, and synonym dictionaries. Only around 3% of incoming strings could be mapped in this manner. We found that character-based GRUs excelled at extracting LOINC elements from noisy text inputs but plateaued around 60% accuracy when combining those elements into a final code. A combination of machine learning and logic-based approaches achieved much higher accuracy and coverage. The resulting hybrid model is shown in Figure 1. A given input string is first processed by six, character-based GRUs for each of the LOINC elements (though the figure shows only three). The outputs of these models are then input to logic that combines them in a final LOINC code. The following sections describe these processing steps and provide a final evaluation.

Name	Unique Possible Values	Unique Values in Data	Coverage
Component	19,507	4,783	25%
Specimen	344	143	42%
Scale	6	6	100%
Timing	668	380	57%
Method	504	212	42%
Property	116	91	78%
LOINC Codes	46,156	11,190	24%

Table 3: Data coverage of possible LOINC values.

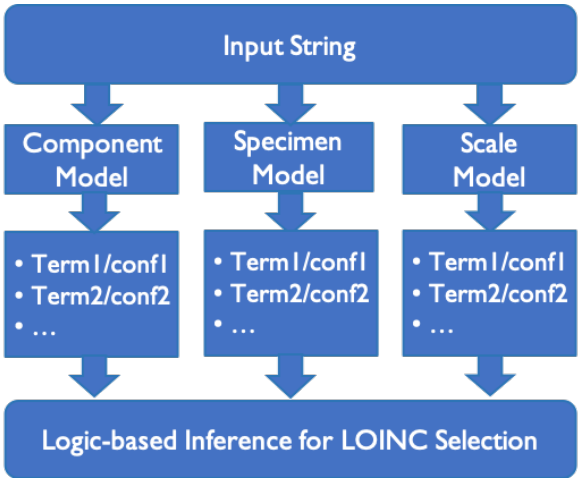

Figure 1: Hybrid solution combining machine learning models with logic for LOINC mapping.

Prediction	Conf
LEUKOCYTE ESTERASE	0.875
LEUKOCYTES ESTERASE NITRITE	0.81
ALBUMIN	0.002

Table 4: Example component predictions for the input string ”Ur Leukocyte Esterase”.

5.1 Extracting Terms with Deep Learning

A character-based GRU was trained for each of the six LOINC elements using the scikit-learn and Keras packages (Pedregosa et al., 2011). The output classes for each element model were the possible values for that element. For instance, the model for the component element was trained to output 19,507 possible classes. A softmax activation function was used with categorical cross-entropy for the loss function. This approach enabled the model to output a probability between 0 and 1 for each of the possible class values. Table 4 shows the top three predictions for the component element given the input string ”Ur Leukocyte Esterase”. Table 5 shows the top predictions for each LOINC element for the same input string.

Element	Prediction	Conf
Component	LEUKOCYTE ESTERASE	.875
Specimen	URINE	.95
Scale	ORD	.86
Method	NONE	.84
Timing	NONE	.856
Property	PRTHR	.763

Table 5: Example predictions for each of the 6 LOINC elements for the input string ”Ur Leukocyte Esterase”.

5.2 Combining Terms with Logic

We hypothesize that several factors contribute to the poor performance of machine learning when predicting a final code from the LOINC elements:

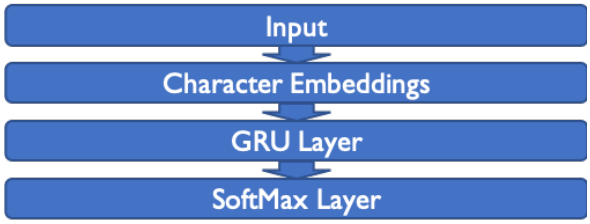

Figure 2: Character-based RNN for extracting a LOINC element.

Class imbalance was severe. While there were sufficient samples of LOINC elements, there were insufficient and imbalanced samples for complete LOINC codes. Machine learning methods are sensitive to class imbalance, whereas logic-based methods are not.

LOINC logic dictates that some elements can be combined while others cannot. For instance, liquid units of measure cannot be combined with specimen that are not liquids. It is easier to explicitly represent these constraints than train models to adhere to them.

Implicit knowledge about the relative importance of LOINC elements is only revealed through conversations with clinical informaticists. For example, clinicians often prioritize the component LOINC element over other elements. A code that has the correct component but incorrect specimen may be acceptable for certain use cases. Because of this implicit prioritization, it is useful to weight elements based on use case rather than taking a completely data-driven approach that treats each element equally in a classifier.

An inference engine was built to combine element predictions into a final LOINC code. The general processing steps are as follows:

1. Start with the highest priority element and generate all candidate LOINC codes that contain the predicted value for that element.

2. Go to the next highest priority element and filter out any candidates that do not have the predicted value for that element.

3. Repeat for all elements, then sort the remaining candidates by a priority weighted average of element confidence values.

For example, the component GRU predicts "Leukocyte Esterase" for the example input string "Ur Leukocyte Esterase". If we start with component as the most essential element, we generate a candidate list of LOINC codes with the predicted value for component: $\{2563-5, 27297-1, 5799-2, 59262-6, 60026-2, 77563-5\}$. We then filter out candidates that do not contain the predicted specimen (or top n predicted specimen). We continue filtering for the rest of the elements in order of priority. Note that it is entirely possible for an input string to be lacking any value for one of the six elements.

Bin	Accuracy %		Coverage %	
	H	R	H	R
$c >.99$	90	71	10	4
$.99 \leq c < 0.75$	85	62	81	63
$.75 \leq c < .5$	80	37	8	23
$c \leq .5$	56	18	2	11

Table 6: Accuracy and coverage percentages binned according to confidence intervals for hybrid and rules-only models. H columns represent hybrid models and R columns represent rules-only models.

5.3 Evaluation

Clinical informaticists average 80% accuracy in a completely manual mapping process. Initial approaches to automate mapping were purely rules-based. Approaches using purely machine learning scored high for element prediction but were less than 70% accurate at predicting final LOINC codes. A hybrid approach combining logic and machine learning provided a dramatic increase in accuracy and coverage. Table 6 shows a comparison. Accuracy metrics are broken into bins based on confidence intervals where c = confidence. Binning was performed to simplify decisions for clinicians. By accepting predictions with a confidence higher than .5, we can achieve human performance of 80% accuracy on a combined coverage of the top three bins or 98% of all incoming custom codes.

5.4 Conclusion

Practical applications of artificial intelligence often require an ensemble of approaches. Combining the multiple approaches can overcome their respective weaknesses in particular use cases. We found that machine learning approaches were best equipped to extract LOINC elements from noisy text inputs, whereas logic-based methods were better at combining those elements into final LOINC codes.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Bora Edizel, Aleksandra Piktus, Piotr Bojanowski, Rui Ferreira, Edouard Grave, and Fabrizio Silvestri. 2019. Misspelling oblivious word embeddings. *CoRR*, abs/1905.09755.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. *CoRR*, abs/1604.00788.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. 2019. Combating adversarial misspellings with robust word recognition. *CoRR*, abs/1905.11268.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *CoRR*, abs/1806.03822.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Jinzhong Xu, Jie Liu, and Ming Liu. 2009. Research on topic relevancy of sentences based on how net semantic computation. volume 2, pages 195–198.

Yue Zhang and Jie Yang. 2018. Chinese NER using lattice LSTM. *CoRR*, abs/1805.02023.