Scaling Up Medical Entity Extraction

Vignesh Venkataraman
Curai and Stanford University
viggy@curai.com

Abstract

In this paper, we present an approach that uses existing medical entity extractors as noisy data labelers for a large corpus of unlabeled text. We show that training a model with multiple noisy labels will produce better performance than a model trained on any individual labeling process. Moreover, we show that deep convolutional neural networks exhibit promise in entity extraction, a task canonically confined to the realm of sequence models.

1 Introduction

Medical entity extraction involves the extraction of medical entities from unstructured text. For example, given an input sentence such as "I have a fever and a headache," an entity extractor would output 'fever' and 'headache' as medical entities. This process of finding medical concepts from text has applications all across the fields of healthcare and artificial intelligence. Particularly pertinent to the intersection of the two aforementioned fields is the process of deriving insights from electronic health record (EHR) data, which is almost unilaterally unlabeled.

Medical entity extractors tend to fall into one of two categories - rule-based or machine-learned. Rule-based extractors are predictable and have no data requirements to build; however, they tend to be rigid and inflexible, and must encode the full spectrum of rules in order to adequately capture the universe of concepts, which is not easily tractable. On the other hand, machine-learned extractors escape the necessity of defining explicit rules, but require large swathes of labeled data in order to build and train. The paucity of well-labeled medical data renders this approach difficult.

2 Approach

Our approach is machine-learned; however, given the lack of properly-labeled medical data, we elect to generate our own noisy labels from unlabeled input text, using other off-the-shelf entity extractors. The specific entity extractors we chose were Apache cTAKES and CliNER from UMass Lowell. These choices are elaborated on in the section below. The labels generated from these entity extractors are obviously not perfect and are inherently biased; in the process of amalgamating them and applying them to a very large dataset, we believe such issues will be overcome. On top of these noisy labels, we apply a variety of models - specifically, logistic regression, multilayer perceptrons, and convolutional neural networks. Finally, we evaluate the trained models on a holdout set of data from the same dataset but with manually curated labels, none of which are used for training.

Practically speaking, a common choice for identifying entities is a Concept Unique Identifier (CUI) from the Unified Medical Language System (UMLS) metathesaurus. This compendium of medical concepts is a superset of almost all of the leading medical ontologies, such as Snomed CT, the Disease Ontology, RxNorm, etc, and is the direct output of cTAKES and a derived output of CliNER. Thus, our end result is a multilabel, multiclass classification problem: given input medical text, output the (many-hot) vector corresponding to the indices of the CUIs present in the input text.

3 Related Work

The approach for this paper mirrors another publication by colleagues at Curai, entitled "Learning from the experts: from expert systems to machine learned diagnosis models" (1). While this paper focused on going from expert systems to machine learned models, the overall approach remains the same: use available resources as labeling mechanisms and train models on top of said labels. In fact, in a way, you can consider properly sourced entity extractors as expert systems themselves.

A paper from Abacha et al. proposes an approach to entity extraction called MeTAE (Medical Texts Annotation and Exploration) (2). This rule-based system first uses an enhanced version of MetaMap to extract medical entities, then leverages semantic knowledge to build connections between the extracted entities to provide accurate annotations to the source text. The work done here illustrates the difficulty of rule-based approaches in problem spaces as vast as medicine. In fact, the paper itself admits: "A limitation to our approach is the fact that it will not always be the case that we have knowledge bases with semantic relationships between medical entities as a starting point."

We also looked at the papers corresponding to the two entity extractors we chose, Apache cTAKES and CliNER. The cTAKES paper (3), by Savova et al. and titled "Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES): architecture, component evaluation and applications", details the design and implementation of the cTAKES entity extractor and examines its performance over five different application areas. The CliNER paper (4), by Boag and Wacome et al. and titled "CliNER: A Lightweight Tool for Clinical Named Entity Recognition", details the design and performance of CliNER, particularly contrasting its approach (two-pass supervised ML, identifying concept boundaries using linear chain CRFs with SVM tagging) with that of cTAKES, which relies on UMLS dictionary lookups.

Finally, we looked at the actual results from the SemEval 2015 Task 14 challenge, which is where our dataset came from (see section below). The in-proceedings report by Elhadad et. al (5) reveals that the SemEval label set is in fact derived only from disorders (or diseases), and does not include any labels for findings, a fact which complicates our evaluation criteria. Nevertheless, the paper is a gold mine of information, illustrating various high-level facts about the dataset and showing that for the task in the challenge that most closely resembles our own, the precision and recall metrics for the top teams were in the range of 70-80%.

4 Dataset and Features

As mentioned above, our dataset is from the HNLP SemEval challenge for the calendar year 2015 (5). The SemEval challenge series is an ongoing evaluation of semantic evaluation systems, which can be compared to the SensEval word sense evaluation series. Task 14 of the challenge in 2015 focused on medical entity extraction, which is of course this paper's primary concern. In particular, Task 14 involved labeling the span and concept identifiers for specific disorders. Because of dataset restrictions, reproducing specific examples of discharge summaries is not allowed, but each discharge summary contains redacted protected health information (PHI), a history of present illness, past medical history, physical exam results, lab values, hospital course, discharge details, etc.

The SemEval-2015 dataset is divided into three main categories: train, development (dev), and unlabeled. Each of these categories is further divided into many different subcategories, describing the type of medical text contained: discharge notes, ECG notes, radiology notes, etc. Due to product priorities at Curai, we chose to focus our experiments on discharge notes, which most closely resemble the types of text we will likely encounter in production settings. The train and dev discharge folders contain 136 and 133 discharge notes respectively, complete with labeled extractions. The unlabeled folder contains 24,144 discharge summaries. A test set exists, but at the time of submission this was only accessible via a Perl script that did not work on the authors' computers.

A critical preprocessing step was to first split the discharge notes into sentences, which helps localize entities to a particular set of tokens. We then ran the unlabeled discharge summaries through the cTAKES and CliNER entity extractors, a task that took over 10 days of nonstop compute time on a 32 CPU machine to complete.

For featurization, we tried a trio of approaches. Initially, we used word count vectorization (via `scikit-learn` (6) APIs to set up our baseline experiments with logistic regression. Word count vectorization outputs a feature vector such that the feature vector's length is the size of the vocabulary, with the value at each index representing a count of the number of times a given word occurred. When we scaled up the size of our training set, this quickly led to memory issues, particularly when converting sparse representations to dense inputs for use with `Keras` (7). To address this, we used a second approach: hashing vectorization. This approach uses the hashing trick to reduce dimensionality in a predictable manner, up to a given number of features (and outputs a vector of that length). We were able to get up to 2^{12} features before again suffering memory pressure. The final approach was to train embeddings for the vocabulary, which we accomplished in conjunction with a convolutional neural network.

5 Methods

As with any ML project, we started with the basic "hello world" approach - attempting to get logistic regression working. In particular, we framed the problem as multiclass, multilabel logistic regression in a one-versus-rest context. The features were culled from each discharge note by running a word count vectorizer on them. For simplicity, our initial experiments used just cTAKES extractions, which directly produced CUI outputs. We cast each CUI label into an index and passed the relevant data into the multilabel logistic regression model.

The next model we tried was a multilayer perceptron (MLP), built using `Keras` (7) against a TensorFlow (8) backend. In particular, we architected a network with two hidden layers, the first of which was a densely connected 2048 node layer and the second of which was a densely connected 1024 node layer. Both layers used the `ReLU` activation function and were followed by dropout layers, at 0.4 and 0.2 drop probability respectively. Finally, we fed the outputs of the last hidden dense layer into an output dense layer, with dimension corresponding to the number of CUIs the model is expected to output. Because of the formulation of the problem as both multiclass and multilabel, each node in this layer is passed through a sigmoid activation and interpreted accordingly, independent of the other nodes. The general formulation for forward propagation through a neural network is given below:

$$z^{[l]} = W^{[l]}a^{[l-1]} + b^{[l]} \tag{1}$$

$$a^{[l]} = g^{[l]}(z^{[l]}) \tag{2}$$

Finally, we also employed a convolutional neural network for this task. This model featured an embedding layer (for learning word embeddings), a number of 1-dimensional convolutions, followed by dropout layers, a flattening layer, and a dense output layer with softmax activations (similar to the MLP above). We tried kernel sizes of 3 and 4 for our filters, and stacked anywhere from 10 to 256 filters in each convolutional layer. We affixed the dropout probability at 0.2 and the embedding size at 256.

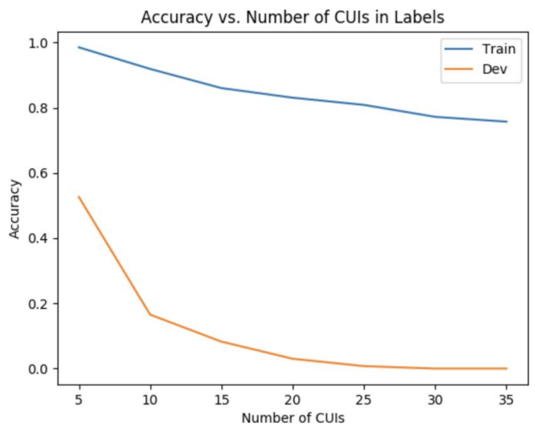
For the latter two models, the loss function was binary cross-entropy loss applied across each of the n-dimensional output neurons - the formula for an individual example is given below:

$$L(y, \hat{y}) = -\sum_{c} y_c \log \hat{y}_c \tag{3}$$

The metric we tracked while training was categorical accuracy, which in `Keras` corresponds to the frequency of predictions matching the actual labels across the entire size of the output dimension.

6 Experiments / Results / Discussion

We initially tried to apply logistic regression, as framed above, on the small train (136 example) set, with validation against the dev (133 example) set. However, the results here were quite odd: while training set accuracy was 1.0, dev set accuracy was a total zero. Initial suspicions centered around a potential lack of overlap in the CUI space; with just 136 training examples and over 4,370 CUIs, the output space is pretty sparse. But further analysis showed that 2,135 of the CUIs were present in both datasets. We then attempted dimensionality reduction by pruning the feature vectors, only including words that were present in over 10% of documents. This too had no effect. Finally, we experimented with greatly reducing the space of CUIs that we needed to classify, first picking randomly to prove the point and then sampling based on CUI frequency. That training chart is presented below:



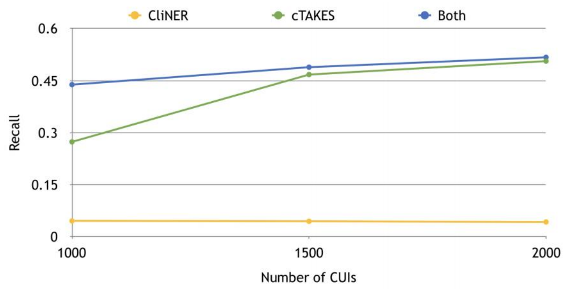
The trendline in this chart indicates that logistic regression is tapped out, in more ways than one - past 10 CUIs, it is unable to produce any sort of reasonable accuracy. Moreover, with such a small training set, and such a vast space of concepts, poor results are almost expected. At this point, we began work on processing all of our data and proceeded to more complex models.

We then ran a series of experiments with the MLP model proposed above, using cTAKES-generated labels, CliNER generated labels, and combined labels from both entity extractors while also varying the number of target CUIs for the model. These results are enumerated below:

Table 1 - MLP Results

Model	Precision	Recall	Validation Categorical Accuracy
MLP (cTAKES, 1000 CUIs)	0.0949	0.2732	-
MLP (cTAKES, 1500 CUIs)	0.1107	0.4672	0.6282
MLP (cTAKES, 2000 CUIs)	0.1137	0.5060	0.6811
MLP (CliNER, 1000 CUIs)	0.0385	0.0453	0.7319
MLP (CliNER, 1500 CUIs)	0.0369	0.0441	0.7425
MLP (CliNER, 2000 CUIs)	0.0358	0.0422	0.6804
MLP (Both, 1000 CUIs)	0.0841	0.4382	0.8321
MLP (Both, 1500 CUIs)	0.1005	0.4885	0.7671
MLP (Both, 2000 CUIs)	0.0949	0.5170	0.7913
MLP (Both, 3000 CUIs)	0.1032	0.5383	0.7672
MLP (Both, 4000 CUIs)	0.1078	0.5471	0.7499

There are a number of key takeaways here. Firstly, we see that adding data from multiple models clearly helps performance. This is particularly obvious across the 1000 CUI experiments. We see that recall for just cTAKES sits at 0.2732, while recall for CliNER is 0.0453. But by combining labels from both extractors, we can get recall all the way up to 0.4382, a textbook example of a whole being greater than the sum of its parts. A second key takeaway is that scaling the number of CUIs the model can output helps performance, particularly in recall. The graph below shows the impact of scaling the number of CUIs across all three model types:



The final takeaway is that precision seems suspiciously low. However, diving into some actual examples, we see that this is easily explainable. In the utilized SemEval evaluation sets (which are actually the train and dev sets in the formal challenge), experts went ahead and hand-labeled the relevant medical entities in a given discharge summary, picking out the location of the concept and the singular CUI that it corresponded to, if any. However, there are two issues here in the context of our usage of these labels. The first issue is that the evaluation sets focus exclusively on *disorders*, or diseases, and thus exclude any findings. For example, "strep throat" would be included in the SemEval labels, where "fever" or "coughing" would not. The second issue is that there are multiple ways to represent the same phrase in terms of UMLS concept identifiers. A perfect example of this is Type I diabetes, which can be represented by the CUIs C0011854 and C1844664. Both of these CUIs are valid for Type I diabetes (they come from different datasets that haven't been de-duplicated, hence the overlap), but the SemEval labels only contain the former CUI, not the latter. Thus, an entity extractor returning both CUIs would have good recall (since it gets the former CUI right) but would suffer a precision penalty, since it is "overpredicting" CUIs. The former issue could be addressed by focusing on disorders only, but that would defeat the practical purpose of this project (which is intended to be a drop-in replacement for an existing entity extractor, which needs to output findings as well). The latter issue could be addressed by not penalizing synonymous CUI representations, but that would require setting up a UMLS installation from scratch and building out a data structure to capture all synonymous CUIs across a very large dataset. Resolving these issues are left as exercises for the future.

Finally, we conducted experiments with convolutional neural networks while also training embeddings for the input vocabulary. These models were quite a bit slower to train because of their depth and parameter scale, and as such experiments here were limited to a pair of runs with just cTAKES labels and a trio of runs with both cTAKES and CliNER labels, with varying vocabulary sizes, kernel sizes, and filter dimensions. These results are summarized in the table below:

Table 2 - CNN Results

Model	Precision	Recall	Validation Categorical Accuracy
CNN (cTAKES, 1000 CUIs, 10 filters, 4 kernel size)	0.0602	0.0918	0.3535
CNN (Both, 2000 CUIs, 6000 vocab size, [256] filters, 4 kernel size)	0.1010	0.4351	0.7498
CNN (Both, 3000 CUIs, 5000 vocab size, [256, 64] filters, 3 kernel size	0.1000	0.3993	0.6705
CNN (Both, 4500 CUIs, Dropout + Elu activation, 6000 vocab size, [64, 32, 16] filters, 4 kernel size)	0.0883	0.3462	0.4899

These results indicate that convolutional network architectures can perform well in the realm of entity extraction, a regime that traditionally lends itself to sequence models. There is still clearly room for improvement, as none of these models were trained for any longer than a few hours due to time constraints. More generally, both the MLP and convnet approaches show that artificial labels, despite being noisy, can still be very useful for training models. Even if the labels are fairly weak in isolation, as is the case with CliNER (which is not pinned to UMLS, and thus doesn't offer CUIs for all the concepts that it identifies), when combining them with other labels, the performance impact is hugely significant. Finally, training embeddings on these networks is also highly tractable, and should hopefully produce valuable insights in future works.

We note that overfitting didn't seem to be an issue between the train and dev sets; if anything, it looks like there is still room to better fit the training data that could be exploited with additional training time and/or hyperparameter tuning.

Across all experiments, we stuck to the default `Keras` parameters for the `Adam` optimizer and left the batch size as 1024 or 2048, depending on the size of the network and the dimensionality of the inputs. Varied parameters included number of features for the hashing vectorizer, the dimensionality of stages of the network, etc.

7 Conclusion / Future Work

In conclusion, this paper presents evidence that suggests using multitudinous sources of noisy labels to train models can achieve good results on a given task, in this case specific to medical entity extraction. In particular, the best MLP and CNN presented in the results section above achieve $> 50\%$ recall on an entirely held-out set of labels, an impressive result given that these labels have no direct relationship with the noisy labels generated by the cTAKES and CliNER entity extractors. Further efforts to tune the performance of the CNN would likely produce even more impressive results.

For future work, there are some logical next steps. Sorting out the evaluation criteria, in particular scoping the precision/recall metrics to disorders and preventing penalties for predicting synonymous CUIs, would likely be the very first place to start. A potential next step would be to add additional entity extractors to the mix as labeling tools; for example, incorporating MetaMap or our own in-house Curai entity extractor would likely add even more value. Another practical step would be constraining the output CUI space even further by instead outputting concepts that are present in our own proprietary knowledge base, which is a strict subset of UMLS concepts that pertain more directly to the products that we are building. Of course, additional feature and model exploration would no doubt yield better metrics, a task which could be automated using many of the automated optimization tools available. Further expansion of training data would also likely improve performance; as it stands now, we are using just an eighth of the full training dataset to optimize for speed of iteration. Finally, no analysis has been done on the embeddings trained for this project. Exploring these embeddings would also be an exercise for the future.

8 Code

The code for this project resides at https://github.com/viggyfresh/cs230-project.

9 Contributions

All implementation work for this project was done by Vignesh Venkataraman. Brainstorming and ideation was assisted greatly by Anitha Kannan and Manish Chablani at Curai. The dataset used for this paper, as well as the GPU resources used for training the multifarious models, were both acquired by Curai.

References

[1] M. Ravuri, A. Kannan, G. Tso, and X. Amatriain, "Learning from the experts: From expert systems to machine learned diagnosis models," *arXiv preprint arXiv:1804.08033*, 2018.

[2] A. B. Abacha and P. Zweigenbaum, "Automatic extraction of semantic relations between medical entities: a rule based approach," *Journal of biomedical semantics*, vol. 2, no. 5, p. S4, 2011.

[3] G. K. Savova, J. J. Masanz, P. V. Ogren, J. Zheng, S. Sohn, K. C. Kipper-Schuler, and C. G. Chute, "Mayo clinical text analysis and knowledge extraction system (ctakes): architecture, component evaluation and applications," *Journal of the American Medical Informatics Association*, vol. 17, no. 5, pp. 507–513, 2010.

[4] W. Boag, K. Wacome, T. Naumann, and A. Rumshisky, "Cliner: A lightweight tool for clinical named entity recognition," *AMIA Joint Summits on Clinical Research Informatics (poster)*, 2015.

[5] N. Elhadad, S. Pradhan, S. Gorman, S. Manandhar, W. Chapman, and G. Savova, "Semeval-2015 task 14: Analysis of clinical text," in *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pp. 303–310, 2015.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[7] F. Chollet *et al.*, "Keras (2015)," 2017.

[8] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: a system for large-scale machine learning.," in *OSDI*, vol. 16, pp. 265–283, 2016.