Radiology: Artificial Intelligence

SPECIAL REPORT

Natural Language Processing of Radiology Text Reports: Interactive Text Classification

Walter F. Wiggins, MD, PhD • *Felipe Kitamura, MD, MSc* • *Igor Santos, MD* • *Luciano M. Prevedello, MD*

From the Department of Radiology, Duke University Health System, Duke University Hospital, Box 3808, 2301 Erwin Rd, Durham, NC 27710 (W.F.W.); Department of Diagnostic Imaging, Universidade Federal de São Paulo, Escola Paulista de Medicina, São Paulo, Brazil (F.K., I.S.); Head of AI, Diagnósticos da América SA (DASA), São Paulo, Brazil (F.K.); FIDI, NESS Health, São Paulo, Brazil (I.S.); and Department of Radiology, Ohio State University, Columbus, Ohio (L.M.P.). Received January 26, 2021; revision requested February 25; revision received April 15; accepted April 22. **Address correspondence to** W.F.W. (e-mail: *walter.wiggins@duke.edu*).

Authors declared no funding for this work.

Conflicts of interest are listed at the end of this article.

Radiology: Artificial Intelligence 2021; 3(4):e210035 • https://doi.org/10.1148/ryai.2021210035 • Content codes: AI IN

This report presents a hands-on introduction to natural language processing (NLP) of radiology reports with deep neural networks in Google Colaboratory (Colab) to introduce readers to the rapidly evolving field of NLP. The implementation of the Google Colab notebook was designed with code hidden to facilitate learning for noncoders (ie, individuals with little or no computer programming experience). The data used for this module are the corpus of radiology reports from the Indiana University chest x-ray collection available from the National Library of Medicine's Open-I service. The module guides learners through the process of exploring the data, splitting the data for model training and testing, preparing the data for NLP analysis, and training a deep NLP model to classify the reports as normal or abnormal. Concepts in NLP, such as tokenization, numericalization, language modeling, and word embeddings, are demonstrated in the module. The module is implemented in a guided fashion with the authors presenting the material and explaining concepts. Interactive features and extensive text commentary are provided directly in the notebook to facilitate self-guided learning and experimentation with the module.

The need for more educational opportunities in artificial intelligence (AI) and machine learning for radiologists and radiology trainees is well established in the radiology literature (1–5). In addition to the development of formal curricula for radiology training programs (6,7), a variety of resources for independent learning can be found in the literature, including many review articles covering general concepts in AI for radiology. In our experience presenting educational content at various radiologic and imaging informatics society meetings, we have found that interactive, hands-on sessions are a great way to foster learner engagement and facilitate understanding of AI concepts through practical examples. Although numerous modules demonstrate concepts in AI for medical image analysis, including the Magician's Corner series in *Radiology: Artificial Intelligence* (8–12), freely available interactive educational content for natural language processing (NLP) in radiology is lacking.

NLP can be defined broadly as a set of methods for processing and analyzing textual data with computers. In the context of radiology, NLP is most often applied to the text from radiology reports, although the general principles can be applied to other textual data extracted from the electronic medical record. In practice, NLP involves a series of steps to convert free-form textual data into structured numeric data that machine learning algorithms can analyze. Potential applications for NLP analysis of radiology reports include information extraction and report classification, machine translation (eg, lay summary generation), and automated impression generation. However, other applications for NLP of radiology reports and/or data from the electronic health record include predictive analytics, coding and billing, labeling of datasets for image algorithm development, cross-sectional imaging series selection, and documentation of critical results and follow-up recommendations. A detailed review of NLP techniques and applications is beyond the scope of this article, and we defer to existing reviews in the literature for further reading on this topic (13–15). However, it is worth noting that substantial progress has been made in the usability of deep learning for NLP in recent years. The evolution of recurrent neural networks (RNNs), such as the long short-term memory architecture, has contributed to substantial gains in NLP performance on medical language tasks (16). The more recent invention of the transformer architecture affords even greater performance for certain tasks and greater sensitivity to contextual variation in semantic meaning of individual words (17). NLP has thus become an even more active and important area for radiology research and clinical applications.

To provide a demonstration of some basic concepts and techniques in NLP, we developed a hands-on, interactive module in Google Colaboratory (Colab). Colab was selected for this demonstration for three primary reasons. First, it is freely available for use and is a familiar entity in the radiology AI community because it has been used in many similar demonstrations, including the Magician's Corner series (8). Second, the platform allows for showing and hiding code cells, which lends itself to demonstrations where participants (eg, noncoders) may be more interested in understanding the methods than understanding the code. Third, the Colab Forms feature allows for easily building in interactive features in which the participants may experiment with certain parameters without having to

Abbreviations

AI = artificial intelligence, MeSH = medical subject heading, NLP = natural language processing, RNN = recurrent neural network, ULM-FiT = Universal Language Model Fine-Tuning

Summary

Authors provide a hands-on introduction to natural language processing (NLP) of radiology reports with deep neural networks. The accompanying Google Colaboratory notebook is designed with noncoders in mind, with the code hidden by default to allow readers to focus on the methods and process of preprocessing text and training an NLP model.

Key Points

- When using deep neural networks to classify radiology text reports, there are important trade-offs to be considered when deciding whether to use the entire text report or a subset such as the "Findings" or "Impression."
- Text must be converted to a numeric format in order to be processed by deep neural networks in a procedure called *embedding* or *language modeling*.
- Adapting or fine-tuning a language model to the radiology "language" can greatly improve the results of text classification models.

Keywords

Neural Networks, Negative Expression Recognition, Natural Language Processing, Computer Applications, Informatics

interact directly with the code and potentially introduce breaking syntax errors. It should be noted that Colab is not a secure platform and not compliant with the Health Insurance Portability and Accountability Act. Thus, protected health information should never be uploaded to or stored in Colab or Google Drive. Alternatives to Colab include other hosted notebook environments, such as Kaggle (*https://www.kaggle.com/code*) and Amazon SageMaker (*https://aws.amazon.com/sagemaker/*), as well as local development tools like Anaconda (*https://www.anaconda.com/*) and Jupyter (*https://jupyter.org/*), among many others.

The data for this module consist of the de-identified radiology text reports included in the Indiana University chest x-ray dataset, available through the National Library of Medicine's Open-I service (18). The code uses the "fastai" deep learning framework (*https://docs.fast.ai*), which is built on top of the PyTorch deep learning library (*https://pytorch.org/*). While the Colab notebook was primarily designed for a guided demonstration, we included extensive documentation and several interactive features to facilitate a self-guided learning experience.

Notebook Setup and Data Preprocessing

The Colab notebook is accessible via: *https://colab.research.google.com/github/wfwiggins/RSNA-NLP-2020/blob/master/Hands_on_NLP.ipynb*. By default, the code in each code cell is hidden. For individuals interested in viewing the code, one can unhide the code by first selecting all cells in the notebook from the "Edit" menu (Edit > Select all cells) or with the keyboard shortcut (Ctrl + Shift + A), then select "Show/hide code" from the "View" menu. Code cells are indicated by the presence of a "play" button that, when clicked, runs the code contained within the cell. The play button appears when the cursor is hovered over the brackets "[]" to the left of each code cell. For beginners, it may be helpful to review the original notebook in its entirety before interacting with the code cells because the expected outputs will be displayed when the notebook is first opened.

Cell 1 sets up the runtime environment by installing the "fastai" libraries and downloading the data. A library called "xmltodict" is also installed for the purposes of preprocessing the downloaded data, which is supplied in the XML format (Fig 1).

Cell 2 parses each XML file and extracts the "Findings" and "Impression" sections from the report along with the report ID. A label of "normal" or "abnormal" is extracted from each report based on the medical subject headings (MeSH) terms provided from the data curation process. Reports containing the MeSH term "Normal" are assigned the "normal" label, while reports that do not contain this MeSH term are assigned the "abnormal" label. While this automated approach is convenient for our demonstration, in practice, human reader review and annotation of reports is often necessary to establish satisfactory ground truth labels for training an NLP model. The results of these preprocessing steps are returned as a "DataFrame" object, which is a tabular data structure from the pandas library (*https://pandas.pydata.org*). Each row in the DataFrame contains the "id," "findings," "impression," "full-text," and "label" for each report. In this case, the "full-text" is the combined text from the report "Findings" and "Impression." The total number of reports in our dataset is 3955.

In the following steps of the module, we explore the data and prepare data to train a model based on the RNN architecture to classify reports as "normal" or "abnormal." The labels we generated thus serve as the training targets for this classifier. RNNs are a class of artificial neural networks that process sequential data, such as sequences of text (reports, literature, DNA sequences) or time-series (videos, stock market prices, etc), while maintaining some internal state that is calculated from prior inputs to the network (Fig 2). A common practice in deep learning is to apply a technique called *transfer learning*, where a model pretrained on a very large dataset is subsequently further trained specifically for a more narrowly defined task—a process referred to as *fine-tuning*. For image classification, one may take a model trained on a large dataset of labeled photographs and fine-tune it on a smaller dataset of chest radiographs to classify images as positive or negative for pneumonia. An analogous case in NLP is to take a model (here, an RNN) that was pretrained on a large corpus of text data (eg, Wikipedia articles) and fine-tune it for a more narrowly defined task (eg, classification of radiology reports as normal or abnormal). As mentioned above, transformers are considered to be the current state-of-the-art deep learning model for NLP tasks; however, given the relatively complexity of the transformer architecture, we chose to carry out this demonstration with an RNN model to make this demonstration and the accompanying explanations more approachable for beginners.

Exploring and Splitting the Data

Cell 3 provides random samples of the data, five entries at a time. Running this cell multiple times should help to provide some

Before preprocessing (XML File)

```
- <Abstract>
    <AbstractText Label="COMPARISON">None.</AbstractText>
    <AbstractText Label="INDICATION">Positive TB test</AbstractText>
    <AbstractText Label="FINDINGS">The cardiac silhouette and mediastinum size are within normal limits. There is no
      pulmonary edema. There is no focal consolidation. There are no XXXX of a pleural effusion. There is no
      evidence of pneumothorax.</AbstractText>
    <AbstractText Label="IMPRESSION">Normal chest x-XXXX.</AbstractText>
  </Abstract>
```

After preprocessing (Pandas DataFrame)

id	findings	impression	full-text	label
1	The cardiac silhouette and mediastinum size are within normal limits. There is no pulmonary edema. There is no focal consolidation. There are no XXXX of a pleural effusion. There is no evidence of pneumothorax.	Normal chest x-XXXX.	The cardiac silhouette and mediastinum size are within normal limits. There is no pulmonary edema. There is no focal consolidation. There are no XXXX of a pleural effusion. There is no evidence of pneumothorax. Normal chest x-XXXX.	normal

Figure 1: Sample of an XML file containing report data (top). The desired information is extracted to a Pandas DataFrame object after preprocessing (bottom).

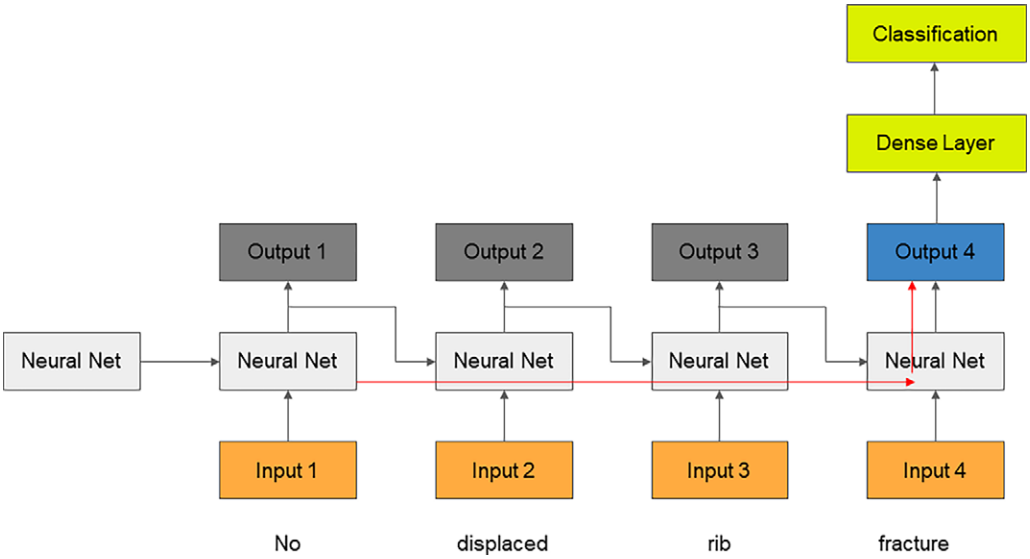

Figure 2: Diagram shows the information flow through a recurrent neural network (RNN). The RNN is presented with each word (or token) in a sequence. Information from prior words in a sentence is fed forward to the network when it receives the next word. This process is repeated until the entire sequence is presented to the network. The red arrows show the flow of prior information when the network is presented with the word *rib*. The output from each step in the RNN is a prediction of the next word in the sequence. When training a text classifier, each output is fed forward to a densely connected layer, which outputs a classification.

insight into the data we will use to train our model. One may see that, typical of variation in radiologist practice, some of the chest radiograph reports contain no separate Findings but include statements of pertinent positive and/or negative findings in the Impression. One may also note that some reports with an Impression similar to "No acute cardiopulmonary disease" are labeled as "abnormal." In some cases, this is due to chronic and/or incidental findings, such as calcified lung nodules or degenerative changes in the spine. This process of data exploration helps inform our approach to modeling because we can see that different inputs to the model (eg, inputting only the Findings or Impression) may affect model performance. A model trained on the Findings will not be able to generate a prediction if the Findings entry is blank. A model trained on the Impression will not be able to differentiate between "normal" and "abnormal" reports that both state "No acute cardiopulmonary disease" in the Impression.

Cell 4 counts the number of "normal" and "abnormal" reports in our dataset and displays this information in a bar chart. This gives us a sense of the class balance (or imbalance) in our dataset. In cell 5, we split the data into a combined training-validation dataset and a held-out test set, which we will reserve for final testing of the trained classification model. The ratio of "normal" to "abnormal" is maintained in these splits by design.

NLP and Language Modeling

Now that we have our dataset split for training and testing, we are ready to proceed with NLP methods to convert our text data into a numeric format so that it may be analyzed by our RNN. The first step in NLP is tokenization, whereby we split each text report into "tokens" or individual chunks of text. A common choice is to tokenize individual words, although one could also choose to tokenize individual sentences, parts of words, or even

individual characters. There are trade-offs with different approaches to tokenization and other text preprocessing methods that may be employed. Subword tokens may help to reduce the size of the "vocabulary" for the corpus by reducing plural and conjugated forms of certain words (eg, "pneumonia" to "pneumon-" + "-ia"; "pneumonitis" to "pneumon-" + "-itis"). Preprocessing steps such as "stemming," where words are reduced to their stems (eg, "pneumonitis" to "pneumon-"), may achieve a similar effect. While reducing the vocabulary size may allow for smaller, and thus less computationally intensive models to be employed, either of these approaches can reduce the semantic information conveyed by these conjugations and compound words. We encourage the reader to consider the possible subword tokenizations of "pneumothorax" and "pneumoperitoneum" and their potential implications for text classification. In this case, we elected to employ word tokenization for its simplicity and to facilitate beginner understanding. Cell 6 demonstrates word tokenization on an example from our dataset.

Our data can be further refined and enriched with further processing. In some cases, we may want to lowercase all words to eliminate the redundancy inherent to capitalization at the beginning of sentences in English grammar (eg, "Pneumonia" becomes "pneumonia"). The "fastai" library provides additional preprocessing features that introduce special tokens. Cell 7 demonstrates the addition of the "xxbos" token to mark the beginning of a text stream and the "xxmaj" token indicates that the subsequent token was capitalized before "fastai" lowercased it.

The next step in NLP is to convert our tokens into numbers, referred to as *numericalization*. Cell 8 computes the "vocabulary" of our training-validation dataset, which consists of the unique tokens found in the dataset. In this case, we see our vocabulary consists of 1192 unique tokens, including the special tokens inserted by the "fastai" library during processing. Cell 9 demonstrates the correspondence between the numeric value and each token in our example text.

At this point, we could simply proceed to train our classifier using these numeric values; however, beyond corresponding to unique tokens, these values contain very little information. Language modeling is a general concept whereby we undertake additional processing to embed some degree of semantic information into our input to the model. While there are many approaches to doing this, the process of training an RNN-based language model to predict the next token in a sequence has proven to be very effective. This method allows the computation of a high-dimensional vector embedded with semantic information for all of the tokens in our vocabulary. Jeremy Howard, the creator of "fastai," and his collaborator Sebastian Ruder, published a method for doing this via transfer learning, which they named Universal Language Model Fine-Tuning (ULM-FiT). In the ULM-FiT procedure, a language model pretrained on a very large dataset of general text (eg, a collection of Wikipedia articles) was then fine-tuned on a narrower dataset, resulting in a model that was more specific to the semantics of the narrower dataset (19). The token embeddings generated by the model can then be used to train a second model to classify text. In the referenced article, this approach was shown to produce state-of-the-art performance (in 2018) for text classification.

In cell 10, we use the ULM-FiT approach to fine-tune a language model pretrained on the Wiki-103 corpus of Wikipedia articles on our training-validation dataset. Most deep learning libraries and frameworks have "model zoos" containing a variety of pretrained models for various NLP tasks (and other deep learning applications), for example, the PyTorch Model Zoo (*https://pytorch.org/serve/model_zoo.html*). Cell 11 saves the vocabulary and embedding layers from the language model for use in the classification model. Figure 3 provides a visual representation of our fine-tuned word embeddings, using the TensorBoard-guided user interface from the TensorFlow library (*https://www.tensorflow.org*) to reduce the dimensionality of the word vectors to a three-dimensional visualization because it would otherwise be very difficult for us to generate an interpretable visualization of a 400-dimensional space. The dimensionality reduction transformation allows us to reduce the number of dimensions, while maintaining some properties of the original data, allowing more interpretable visualizations of the data. Semantically similar words are clustered closer together in the resultant vector space. The "cosine similarity" metric calculates the distance between any two of the original, high-dimensional vectors. This provides another means of visualizing the most semantically similar words to the queried word without employing dimensionality reduction.

Another way to better understand word embeddings and vectors is with word analogies and vector arithmetic. The classic example of this is the "woman is to queen as man is to ?" example often given with explanations of the "Word2Vec" algorithm for generating word embeddings (20). In Figure 4, we provide an example of this in the context of the embeddings derived from fine-tuning the language model on the corpus of chest radiograph reports used in this demonstration. When reformulated as a word vector equation, one can gain some insight into the semantic information embedded into the language model.

Training and Evaluation of the Text Classifier

Cell 12 downloads the fine-tuned embeddings from GitHub—a step designed to allow the reader to skip the fine-tuning steps in cells 10 and 11 because they may take several minutes to complete.

Cell 13 sets up the training experiment. The Google Colab Forms feature is exploited to allow the learner to experiment with different training targets (ie, "full-text," "findings," or "impression"), different percentage splits of the validation dataset (set to 0.3 or 30% by default), and using the fine-tuned "radiology" language model embeddings versus the pretrained embeddings (from the Wikipedia corpus). The combined training-validation data are then randomly split into training and validation sets by the "fastai" function that creates "TextDataLoaders," which serve the data to the GPU in batches. A sample batch of the training data is then displayed in the format in which it will be presented to the model on the GPU for training.

In cell 14, we first train the final layers of the model, which do not have pretrained weights. This is a standard technique employed in transfer learning, called *fine-tuning*, and is done to avoid propagating error back through the higher layers in the network during the initial cycle or "epoch" of training. The

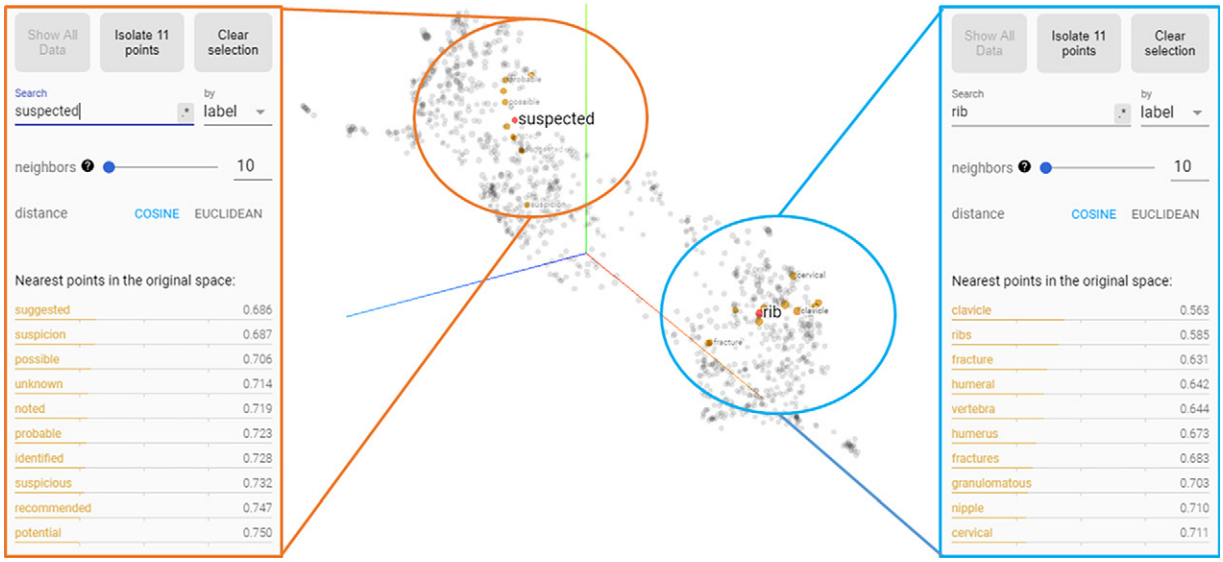

Figure 3: Visualization of word embeddings generated by fine-tuning of the language model. After applying dimensionality reduction in Google TensorBoard, the resultant word vectors are plotted in a three-dimensional graph. The nearest neighbors in the original high-dimensional vector space are determined by their cosine similarity to the target word vectors "suspected" and "rib." Smaller cosine similarity scores correspond to more semantically similar words.

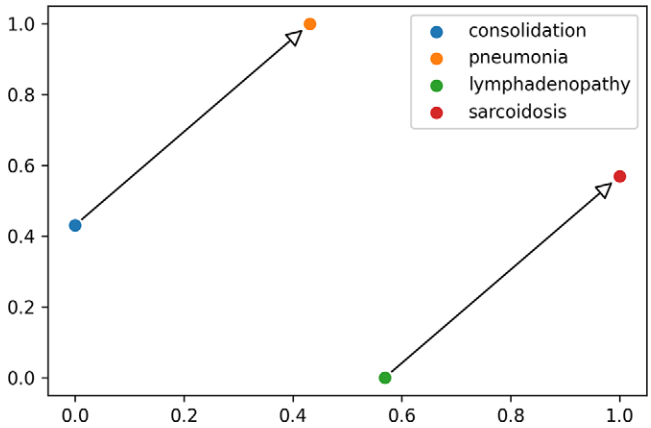

Figure 4: Consider the word analogy problem: "consolidation is to pneumonia as lymphadenopathy is to *x*?" Using word embeddings (or vectors) and vector arithmetic, we reformulate the problem as f(pneumonia) − f(consolidation) = f(x) − f(lymphadenopathy) where f(word) is the vector for a given word. Using the embeddings from the fine-tuned language model and solving for f(x) yields the word embedding for "sarcoidosis." When the word vectors are reduced to a two-dimensional plot, we see that the distance and trajectory between the vectors for the pairs (consolidation, pneumonia) and (lymphadenopathy, sarcoidosis) are approximately equal.

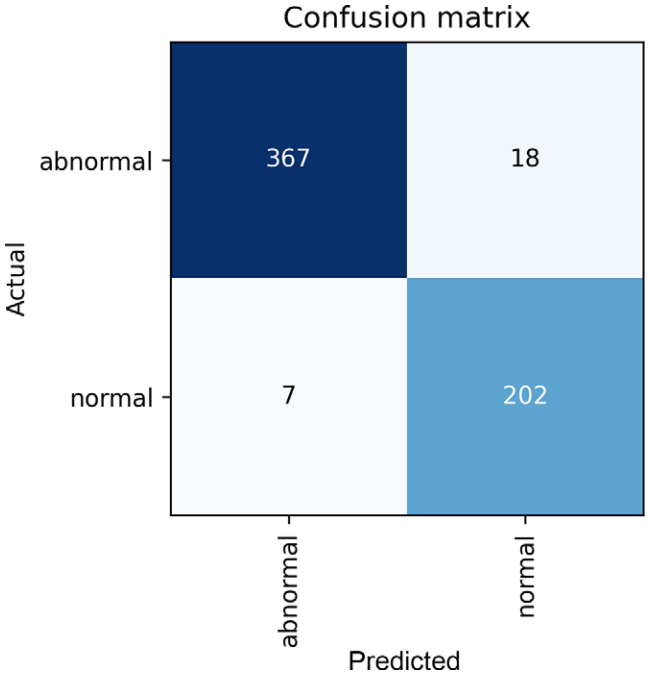

Figure 5: Example confusion matrix generated in the Google Colab notebook using the default values for the training experiment. Upper left: true positives; lower right: true negatives; lower left: false positives; upper right: false negatives.

subsequent lines of code progressively "unfreeze" (ie, make trainable) and train the higher layers of the network. The training and validation "loss" are output below, along with the accuracy on the validation set. The loss is a measure of how far off the model's predictions are from the target labels. This is what the model uses to update the weights in each layer of the network. We should see our validation loss continuously decreasing. When this metric begins to increase, it is a sign that we are beginning to "overfit" to our training data and further training may hinder the model's performance on new data (eg, our held-out test dataset).

Cell 15 loads the held-out test dataset into a "TextDataLoader" and then uses the "ClassificationInterpretation" object from the "fastai" library to run our trained model in inference mode on the test data and generate reports on our model's performance. The output of this cell is a table of performance metrics on the test dataset, including performance metrics on each of our classes (ie, "normal" and "abnormal"). To elaborate on these metrics, we

will consider only the "abnormal" class to be the "positive" result. Precision (ie, positive predictive value) gives the percentage of reports that are truly "abnormal" out of all reports predicted to be "abnormal" by our model. Recall (ie, sensitivity) gives the percentage of reports predicted to be "abnormal" out of all "abnormal" reports. The F1-score, defined as the harmonic mean of precision and recall, is a measure of overall diagnostic accuracy commonly used in machine learning. Comparing the performance of your model on the test dataset to the performance on the validation dataset (supplied in the output from cell 14) can help one refine the training procedure to improve model performance. For example, if the model accuracy on the test dataset is significantly lower than the output on the validation dataset, the model may be "overfitting" to the training and validation data, in which case you might consider training your model for fewer epochs, decreasing the learning rate, or unfreezing fewer layers of the model during fine-tuning.

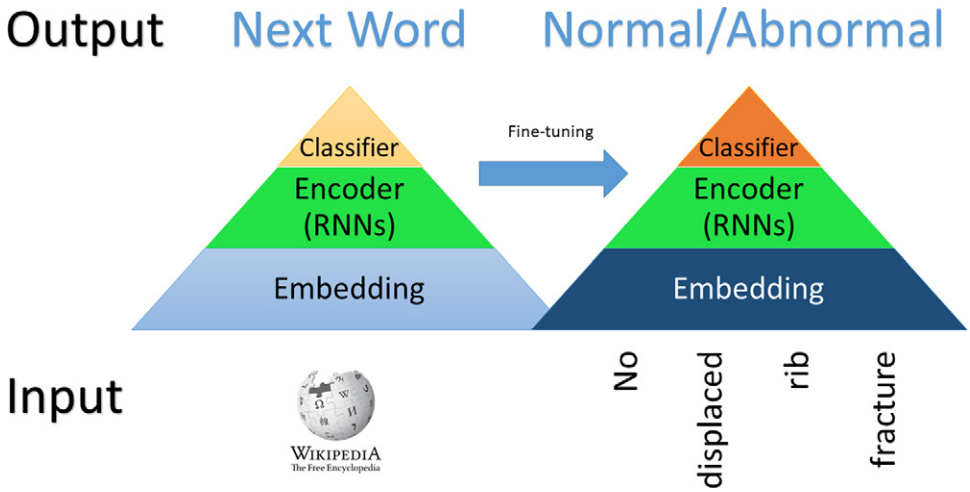

Figure 6: Graphical summary of the Universal Language Model Fine-Tuning approach. On the left-hand side of the figure, the language model takes text input and is trained to predict the next word in the sequence. After fine-tuning the language model on text from radiology reports, the embeddings from the language model are applied to a text classification model trained to predict whether the report is "normal" or "abnormal."

In cell 16, we plot a confusion matrix, which gives us a visual indication of our model's performance (Fig 5). The cells of the matrix that are off of the main diagonal indicate false positives and false negatives. Understanding where your model fails is a very important step in deciding whether to undergo further development and how the model will ultimately be implemented. Another way to analyze your model's failures is to view the "top losses," which represent the test data points on which your model is the furthest off from the target label. This approach is taken in cell 17. One may notice that some of the reports have been seemingly inappropriately labeled as "abnormal" by our automated approach to establishing the ground truth labels for this dataset from the MeSH terms supplied. In analyzing model failures, it is often helpful to review the original data for misclassified reports to ascertain why a given label was assigned in the annotation process, whether it was appropriate or accurate, and why the model may have misclassified the report. It is possible that such review could uncover multiple instances of mislabeled data that could have been prevented by human reader annotation of reports.

We have now completed the ULM-FiT procedure for fine-tuning a language model to generate task-specific word embeddings for subsequent use in a text classification task. A graphic summary of this approach is provided in Figure 6. As we discussed earlier, the language model we employed to generate our embeddings is designed to predict the next token in a sequence. To provide another way of understanding how the ULM-FiT approach produces better performance, cell 18 provides an interactive Form for experimenting with different inputs to either the pretrained or the fine-tuned language model. After selecting the inputs, running the cell will produce sentences of a specified length generated by the specified language model. One should see that the fine-tuned model produces more realistic text, similar to the reports in our database. It can thus be inferred that the embeddings produced by this model convey more semantically accurate information to our classification model.

Conclusion

In this article, we used the "fastai" library to preprocess radiology text reports for input into two neural NLP models based on the RNN architecture. We then fine-tuned a language model on our reports to generate better word embeddings, which were subsequently used to train a text classification model to predict whether a chest radiograph report is normal or abnormal. We hope this hands-on article will serve as a guide for both coders and noncoders to understand this basic pipeline for training a deep learning NLP model to classify free-text radiology reports.

Author contributions: Guarantors of integrity of entire study, W.F.W., F.K.; study concepts/study design or data acquisition or data analysis/interpretation, all authors; manuscript drafting or manuscript revision for important intellectual content, all authors; approval of final version of submitted manuscript, all authors; agrees to ensure any questions related to the work are appropriately resolved, all authors; literature research, W.F.W., I.S., L.M.P.; experimental studies, W.F.W., F.K., L.M.P.; and manuscript editing, all authors

Disclosures of Conflicts of Interest: W.F.W. disclosed no relevant relationships. **F.K.** Activities related to the present article: disclosed no relevant relationships. Activities not related to the present article: author is consultant to MD.ai; employed by DASA. Other relationships: disclosed no relevant relationships. **I.S.** disclosed no relevant relationships. **L.M.P.** Activities related to the present article: associate editor of *Radiology: Artificial Intelligence*. Activities not related to the present article: disclosed no relevant relationships. Other relationships: disclosed no relevant relationships.

References

1. Rubin DL. Artificial Intelligence in Imaging: The Radiologist's Role. J Am Coll Radiol 2019;16(9 Pt B):1309–1317.
2. Simpson SA, Cook TS. Artificial Intelligence and the Trainee Experience in Radiology. J Am Coll Radiol 2020;17(11):1388–1393.

3. Slanetz PJ, Daye D, Chen PH, Salkowski LR. Artificial Intelligence and Machine Learning in Radiology Education Is Ready for Prime Time. J Am Coll Radiol 2020;17(12):1705–1707.
4. Tajmir SH, Alkasab TK. Toward Augmented Radiologists: Changes in Radiology Education in the Era of Machine Learning and Artificial Intelligence. Acad Radiol 2018;25(6):747–750.
5. Wood MJ, Tenenholtz NA, Geis JR, Michalski MH, Andriole KP. The Need for a Machine Learning Curriculum for Radiologists. J Am Coll Radiol 2019;16(5):740–742.
6. Lindqwister AL, Hassanpour S, Lewis PJ, Sin JM. AI-RADS: An Artificial Intelligence Curriculum for Residents. Acad Radiol 2020. 10.1016/j.acra.2020.09.017. Published online October 15, 2020.
7. Wiggins WF, Caton MT, Magudia K, et al. Preparing Radiologists to Lead in the Era of Artificial Intelligence: Designing and Implementing a Focused Data Science Pathway for Senior Radiology Residents. Radiol Artif Intell 2020;2(6):e200057.
8. Erickson BJ. Magician's Corner: How to Start Learning about Deep Learning. Radiol Artif Intell 2019;1(4):e190072.
9. Erickson BJ. Magician's Corner: 2. Optimizing a Simple Image Classifier. Radiol Artif Intell 2019;1(5):e190113.
10. Erickson BJ, Cai J. Magician's Corner: 4. Image Segmentation with U-Net. Radiol Artif Intell 2020;2(1):e190161.
11. Erickson BJ, Cai J. Magician's Corner: 5. Generative Adversarial Networks. Radiol Artif Intell 2020;2(2):e190215.
12. Huber NR, Missert AD, Erickson BJ. Magician's Corner: 7. Using Convolutional Neural Networks to Reduce Noise in Medical Images. Radiol Artif Intell 2020;2(5):e200036.
13. Cai T, Giannopoulos AA, Yu S, et al. Natural Language Processing Technologies in Radiology Research and Clinical Applications. RadioGraphics 2016;36(1):176–191.
14. Chen PH. Essential Elements of Natural Language Processing: What the Radiologist Should Know. Acad Radiol 2020;27(1):6–12.
15. Pons E, Braun LM, Hunink MG, Kors JA. Natural Language Processing in Radiology: A Systematic Review. Radiology 2016;279(2):329–343.
16. Ong CJ, Orfanoudaki A, Zhang R, et al. Machine learning and natural language processing methods to identify ischemic stroke, acuity and location from radiology reports. PLoS One 2020;15(6):e0234908.
17. Bressem KK, Adams LC, Gaudin RA, et al. Highly accurate classification of chest radiographic reports using a deep learning natural language model pretrained on 3.8 million text reports. Bioinformatics 2020. 10.1093/bioinformatics/btaa668. Published online July 23, 2020.
18. Demner-Fushman D, Kohli MD, Rosenman MB, et al. Preparing a collection of radiology examinations for distribution and retrieval. J Am Med Inform Assoc 2016;23(2):304–310.
19. Howard J, Ruder S. Universal Language Model Fine-tuning for Text Classification. ArXiv 1801.06146. [preprint] https://arxiv.org/abs/1801.06146. Posted January 18, 2018. Accessed January 2021.
20. Allen C, Hospedales TM. Analogies Explained: Towards Understanding Word Embeddings. ArXiv. 1901.09813. [preprint] https://arxiv.org/abs/1901.09813. Posted January 28, 2019. Accessed March 2021.