TongAI: Helping Neuroradiologists Do Better Things

Nikita Demir
Stanford University
ndemir@stanford.edu

Jose Giron
Stanford University
jmgiron@stanford.edu

Abstract

Neuroradiologists spend a large portion of their time recommending an imaging protocol based on a doctor's patient description. In conjunction with the Stanford Hospital, we compile a novel dataset and implement state of the art Deep Learning NLP techniques to automatically make the protocol recommendation. We find that a combination of FastText embeddings and a custom FastText model variant provide great overall results. Our research indicates that this is the first time someone has attempted to automate this task.

1 Introduction

Currently, when a doctor wants to request imaging studies after examining a patient, the examining physician will fill out a form online. Here, the doctor describes the patient (age, gender) and includes the reason for the study in the form of a patient description. A radiologist will then examine this request, determine what the right protocol for the case is, and then schedule that procedure. This entire process is time-consuming and most of the time, fairly straight-forward. The radiologist's expertise is only really needed on a handful of cases where the actual protocol to be run is not clear based on the doctor's description or the nuances of the case require an expert's knowledge. We seek to automate this process using Deep Learning Natural Language Processing techniques so that radiologists' can focus their attention on the cases that are really needed.

We found the best results by using our own modified version of the FastText model published by Joulin et al.[2] which we call TongAI and using a novel document representation method described by Dubois et al.[5] in conjunction with finetuned pretrained embeddings available online from FastText.

2 Related Work

To the best of our knowledge, this paper is the first one to target the specific task of classifying requests into specific imaging protocols. This process is currently done by trained radiologists, and probably hasn't been able to capture much attention before. However, we based on approach on existing text classification approaches, as well as methods to efficiently represent clinical text.

Much of our approach is based on FastText, which actually consists of two parts, one for word embeddings and the other for efficient text classification.

The first FastText paper is by Bojanowski et al.[1] at Facebook research. Their model is basically the skipgram model introduced by Mikolov et al.[3] but instead of treating words as atomic structures, it treats words as bag of n-grams. For example, if we were using character 3-grams, the word "where" would become the set "<wh", "whe", "her", "ere", "re>", where the tags "<" and ">" are appended at the beginning and end of the word. Each of these 3-grams then has its own embedding, with the embedding for "where" being the sum of these. This means the model can learn intra-word structures to derive the meaning of words. This seemed specifically pertinent in the medical field, given the importance of word roots in understanding word meaning (ex: the root 'neur' indicates something is related to the brain).

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

The second FastText paper is by Joulin et al. [2] and it introduces a couple tricks for efficient text classification. This model averages the word embeddings in a document to create a representation of a document that can then be passed through a neural network. The paper also suggests using word n-grams to deal with the bag-of-words model's invariance to word order. They pass this document representation through a single linear layer and then calculate the probability distribution across classes using heirarchical softmax. They use heirarchical softmax to increase efficiency when the range of output classes is too large. They finally recommend using cross-entropy loss as the objective function, which is the standard in text classification.

To represent the text part of the imaging requests we looked at two papers. The first by Wang et al.[4] was a comprehensive review of different pre-trained embeddings for biomedical natural language processing using both intrinsic and extrinsic evaluations. They conclude that while domain specific embeddings help achieve greater accuracy in experiments, this difference isn't significant. They hypothesize that this is because the general pretrained embeddings are trained on much larger datasets, which allows them to capture more information about the English language than the domain specific embeddings that are trained on smaller datasets.

The second paper on text representation was by Dubois et al.[5] and it compared two methods of representing free-form clinical text to allow for transfer learning. They train two models on a large set of unlabelled data from the Stanford Hospital, and then apply these to a target task with a much smaller dataset. They prove that their methods increase performance on target tasks. They investigate two methods to represent clinical text. The first one is an embed-and-aggregate method. They develop embeddings using a skipgram model. First, they embed all words in an input document. Then, they create a representation of that document by creating three intermediate representations by taking the mean, min, and max across the embedding dimensions and then concatenating this to create a final representation. They use this fixed-length representation as the input for the target task. They also try training an RNN to create document representations, to good effect. However, we were unable to find the pretrained weights for this last trained models, so instead just use the embed-and-aggregate approach described here.

Our approach was inspired by these papers, and will be described in the subsequent section. However, we also experimented with variations of these approaches to get to the Tong AI model.

3 Approach

3.1 Naive Bayes

Our problem fits into the general NLP category of text classification. As a baseline step we decided to first implement multinomial Naive Bayes classifier. Naive Bayes is a standard and often pretty powerful text classification algorithm that computes the probability for each class based on an application of bayes rule to the input:

$$p(C_k|x) = \frac{p(C_k)p(x|C_k)}{p(x)}$$

Where x is the vector input and C_k is one of the k classes. The denominator is unimportant as we are only interested in the comparison of probabilities between classes and it is the same for every class. The numerator can be split up using a "naive" conditional independence assumption between elements of x to arrive at the following overall formula:

$$p(C_k|x) \propto p(C_k) \prod_{i=1}^{n} p(x_i|C_k)$$

Where $p(x_i|C_k)$ is computed per word. We take the log of this quantity to prevent numerical underflow and finally incorporate Laplace Smoothing for cases when a class and word did not appear together in the training data. Finally, we multiply by the multinomial counts of a word in the input sentence.

Naive Bayes can perform very strongly in text classification problems since often the combination of word probabilities is enough to point to a specific class if there are certain highly relevant keywords.

Our input text also lacks the kind of structure that often confuses Naive Bayes ("not good" in sentiment classification for example) and so we theorized that Naive Bayes may actually end up performing state of the art. Fortunately, despite Naive Bayes performing very well as a baseline, through methodical and extensive architecture exploration we were able to substantially improve on it.

3.2 CNN

Our first deep learning approach was to implement a simple CNN on top of our word embedding layer motivated by Yoon Kim [6]. We padded the output of our word embedding layer with zeros to arrive at a fixed size matrix of sentence length $\times$ embedding dim. We then used several 1-dimensional filters of different sizes to capture n-gram representations and maxpooled their outputs. Finally we passed the pooled output through a single dense layer. As this model did not produce better results than Naive Bayes we did not pursue it more.

3.3 FastText

Our main approach was based on the model described by Joulin et al.[2] and the clinical text respresentations described by Dubois et al.[5]. We choose the FastText approach because we noticed that a lot imaging requests were fairly unstructured. Mostly, doctors would just note down an array of symbols and ask for some kind of imaging. Since syntactic structure wasn't that important and instead a lot of the analysis could be done by just capturing some of the key words in each document, it seemed that a simpler model could do well compared to more complex models that do capture structure, like RNNs. This is also the reason we think Naive Bayes did so well. We also wanted a model that was lightweight, and could run in any computer. Our goal was that this model could be used in any hospital, and given the difficulty in training more complex models, we wanted something that could be trained quickly on a cpu.

Our model first tokenizes each of the requests of imaging using spaCy, an industrial strength tokenizer for the english language. We then generate bigrams based on these representations, which we concatenate at the end of each sentence. Bigrams allow for some data localization information, which is found to increase performance by Joulin et al.[2]

We then create embeddings for each of the terms from the previous step. We tried multiple different embeddings, but ended up getting the best results using the 300 dimensional pre-trained weights from FastText available through TorchText.

The FastText embeddings are trained using the unsupervised skipgram model with negative sampling introduced by Mikolov et al. [3]. This model is inspired by the distributional hypothesis and attempts predict words that appear in the context of a target word. To do achieve this, this model maximizes the following equation

$$\sum_{t=1}^{T} \sum_{c \in C_t} \log p(w_c|w_t),$$

where w_c represents the context words around the target word w_t. To define the probability p of a word appearing in the context of each other, softmax is used, where a scoring function s is assumed to be given:

$$p(w_c|w_t) = \frac{e^{s(w_t,w_c)}}{\sum_{j=1}^{W} e^{s(w_t,j)}}$$

The only change to that Mikolov et al.[3] model is that a new score function is necessary to adapt to the bag of n-grams scenario. The score function is:

$$s(w,c) = \sum_{g \in G_w} z_g^T v_c$$

where G_w is a set containing the character n-grams in the target word, z_g is the embedding for the character n-gram g, and v_c is the potential context word. This formula is taken from Bojanowski et al. [1].

We also tried training out own embeddings using the skipgram model available at fasttext.cc, but saw limited results. The embeddings we trained were 100-dimensional and had character 3 grams, 4 grams, and 5 grams and were trained on our train dataset. Our motivation for using custom-made FastText embeddings was that after talking to Elizabeth Tong, our mentor, she mentioned that each hospital uses a slightly different language around imaging procedures and therefore decided to try the FastText method to make our own embeddings. We implemented an EmbeddingBag class based on an implementation from FastText which converts a string of words to it's corresponding set of FastText embeddings given character n-gram embeddings.

We ended up abandoning this approach after finding that Glove-embeddings provided significantly better results that our pre-trained embeddings even with a smaller neural network built on it. This is in-line with the findings from Wang et al.[4], which conclude that using domain specific embeddings provide limited but not significant results. We believe that our trained embeddings provided valuable information about each of the medical terms, but lacked in terms of the quality of general English terms. There are also benefits from having access to subword embeddings, since it allowed us to create embeddings for unseen words (like mispelled words). However, because our model used an EmbeddingBag based on a character n-gram model with around 2 million n-grams, the model didn't parallelize well and was extremely computationally intensive to train despite the relatively simple neural network we built on top of it.

After getting embeddings for each of the words in our document, we represent it by concatenating the mean, max, and min of all the embeddings in the document. Our original approach was based on Joulin et al.[2] and consisted of calculating the average of all the embeddings (ignoring padding elements). However, we decided to change our representation based on Dubois et al.[5]. Dubois et al.[5] proposes the approach which consisted of taking the min, max, and mean of the word embeddings for each document and then concatenating these 3 representation into a vector that was 3 times the embedding dimension. We also calculate mean differently than the traditional pytorch mean function. This was done to ignore the paddings introduced to allow for batches of documents to be calculated in parallel. Since each document had a different length, we had to adjust mean to recognize that. In our case, since we ended up using 300 dimensional embeddings, our document representation had 1200 dimension.

We concatenate into this 1200 dimensional vector the age and gender of the patient. The age is bucketized an 11-dimensional vector one-hot vector, where each value represents a decade. Gender is represented as a binary value. We decided to use age and gender because we found a difference in the distribution of protocols across ages and genders. This is described more in depth in the data section below.

Then, as described in Joulin et al.[2], we pass this 1212 dimensional representations through one linear layer, and then calculate the log-softmax. When making predictions, we simply choose the maximum from this distribution because log is a monotonically increasing function. We experimented with different numbers of hidden layers, but found the best results from using only a single layer.

Finally, as out objective function we minimize negative log likelihood loss using the pytorch implementation. This loss is defined as follows:

$$\mathcal{L}(\hat{y}, y) = -\sum_{c=1}^{C} \hat{y}_c y_c$$

Where C is the total number of classes, which is 11 in our case. Note that the input $\hat{y}$ is the log of the softmax output, so $\hat{y}_c$ meant to approximate $log(p(X_i = c))$, the probability that the example ith example equals any given class c. Therefore, our loss function is equivalent to cross-entropy loss, which is meant to maximize the probability of the right output class in a softmax output.

4 Experiments

4.1 Data

In conjunction with the Stanford Hospital we created a novel dataset from electronic medical records over the past ten years. Medical data is often very hard to come by and so we were thankful for Dr.

Tong's help in communicating with the hospital IT department to acquire the initial data file. Imaging study recommendation can be seen as two different classification tasks. The first task is to determine the region that needs to be imaged: Brain, Head Neck, L-Spine, C-Spine, etc. And the second task is given that region to determine the specific protocol to use. After discussing with Dr. Tong we decided to focus on classifying a particular region and settled on pursuing the Brain imaging studies first. The methods we develop can in the future be scaled up across different regions.

The dataset contains the anonymized patient id, reason for imaging as well as the patient's age and sex (our inputs), and the neurologist's protocol recommendation (our target output). Medical data can often be noisy and inconsistent. We discovered that the dataset contained many entries unrelated to our task (contrasting agent recommendations for example) where a protocol should have been entered. It seems that the hospital changed their internal record keeping structure recently leading to this confusion. We removed the problematic entries and then further preprocessed the data to remove duplicates and entries that were missing protocol recommendations or patient descriptions. From the original 261,585 classes we were left with just 13,075 well formatted data entries for the Brain protocol recommendations task. Dr. Tong also determined that there seems to be a labelling around 10% of cases, which caps the achievable performance. Although our final dataset was not balanced between the 11 Brain protocol classes, we decided it was not unbalanced enough to prevent our models from learning well.

We randomly shuffled and split our dataset with an 80 / 20 dev/test split to maximize our already constrained training data size.

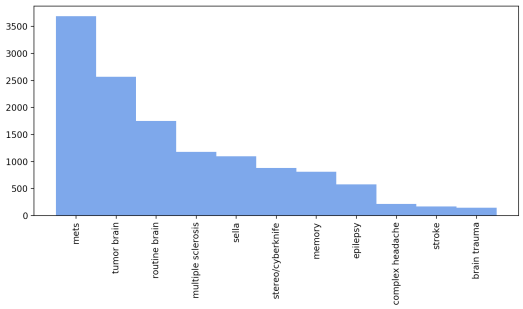

Figure 1: Overall dataset distribution

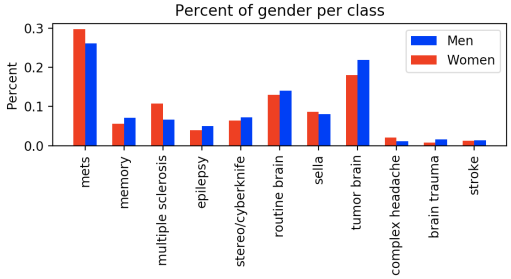

Figure 2: Gender distribution over classes

Table 1: Data examples

Age	Gender	Reason for imaging	Protocol
86.0	f	progressive dementia	routine brain
49.0	f	49 yo female with h/o pituitary mass.	sella
69.0	f	surgical planning. please include t1 and t2 with fiducials.	stereo/cyberknife
69.0	m	lung mass evaluate for metastatic disease	mets
47.0	m	follow up for ms	multiple sclerosis

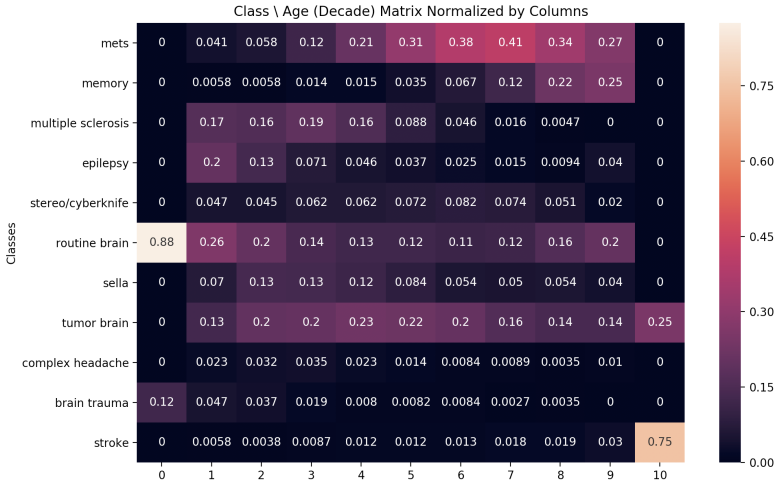

Figure 3: Class versus age in decades normalized by age. Shows how age helps determine class.

4.2 Evaluation Metric

Accuracy is the standard logical choice for evaluating a classification metric. It is defined as the percent of examples classified correctly. Although this can often be problematic in unbalanced datasets, we determined that our dataset was not dominated by any one class enough that accuracy reports would be misleading. We utilized both overall accuracy for rapid model iteration as well as per class accuracy for error analysis. In addition, since we are hoping to use our algorithm to assist doctors rather than outright replace them, we realized that what we really wanted to focus on was high recall. We want our model to retrieve a large portion of the relevant documents because we may in the future show the doctor a couple possible options and let them choose the correct one so we would like the relevant document to be in the options. Recall is defined as the percentage of a class correctly retrieved. We also looked at precision for interest as well as F-score as an average of precision and recall. For precision, recall, and F-score we first calculated the per class values and then averaged them over the classes by weighting by the number of true occurrences of each class to arrive at a more holistic metric over our class imbalance. Due to our large class number we also visualized confusion matrices to better understand during error analysis our model's mistakes.

4.3 Experimental Details

We first wanted to examine how well random guessing would perform on our unbalanced dataset. We built a class distribution from our training data and then sampled according to that distribution during test time. We next examined how useful just the age and just the gender were by building simple feedforward two dense layer neural networks.

We ran Naive Bayes to achieve a preliminary baseline and gauge where the difficulties would lie with our dataset. For all approaches, we tokenized our sentences to get separate word terms. For Naive Bayes we experimented with a simple multinomial vector of counts as well as tf-idf weights. For our following deep learning approaches we experimented with a variety of embeddings from training our own using the FastText approach to downloading pretrained GloVe and FastText ones and finetuning. We found the best approach to be a combination of pretrained FastText 300d embeddings with further finetuning during the training process.

Our initial Deep Learning approach was to use a Inception-like one layer CNN with filter sizes of 1, 2, and 3. We used 100 filters for each size and then passed the output through a ReLU activation followed by a 1D max pool operation, finally we passed it through a dropout layer ($p = 0.5$) and a densely connected layer and took the log softmax. We early stopped the model after 6 iterations.

For our final FastText variant model we experimented with many different configurations and architectures. We found that generally we were dealing with a variance problem as our model

was perfectly capable of overfitting given enough epochs. To combat this our main resource was early stopping. We also experimented with adding Dropout before the dense layers and adding L2 regularization with different values for lambda. We extensively searched for ways to lower our variance, but due to the inefficacy of common variance-reduction methods and the large extent of our architecture search we came to believe that our validation results were approaching an upper bound determined by the size of our dataset. We experimented with adding more dense layers, trying 5 / 20 / 100 for hidden layer sizes, training dense hidden layers for the age and gender separately before concatenating, and different ways of performing the pooling operation. We also experimented with stopping word embedding fine-tunning after 6 epochs to minimize the amount of overfitting in combination with adding more hidden layers to the model to try and eek out more meaning.

We trained all of our deep approaches using an Adam Optimizer with learning rate equal to 3e-3 and betas of 0.9 and 0.999. We tuned the learning rate to achieve faster convergence but since our main FastText model trained quite quickly (a minute or so, roughly 5 epochs) it was not a great concern.

4.4 Results

Table 2: Results

Model	Accuracy	Precision	Recall	F-Score
Random	15.45%	0.170	0.155	0.155
Age	31.79%	0.195	0.318	0.229
Gender	28.35%	0.089	0.284	0.134
Naive Bayes	76.73%	0.777	0.767	0.760
CNN	64.20%	0.642	0.639	0.633
Vanilla FastText	78.60%	0.790	0.786	0.778
TongAI	**82.06%**	**0.829**	**0.821**	**0.816**

The impressiveness of our TongAI results surprised us. Not only were we able to achieve pretty impressive accuracy and recall, but we also significantly improved upon Naive Bayes through our extensive architecture and hyperparameter search. The common denominator we saw across our models was that given enough epochs a model would overfit to the training word distribution because of our dataset's small size.

For our suggested use case these results are more than enough, especially given the high recall and the error intrinsic in the dataset.

5 Analysis

As the confusion matrix in Figure 4. shows our model performs pretty well across the large classes but less well with the three smallest classes 'complex headache', 'brain trauma', and 'stroke'. Interestingly all three of these are misclassified to the 'multiple sclerosis' protocol. Without greater medical knowledge it is hard to say why 'multiple sclerosis' is the predicted target and analysis of the misclassified data examples didn't provide any hints.

As the random model results and confusion matrix show our dataset imbalance did not lead to misleading accuracy. Our TongAI model was able to learn pretty well across the classes it had enough data for and did not just predict the most common class.

Our results for the larger classes are even more impressive given that Dr. Tong identified a roughly 10% error in her analysis of the original dataset. While this was a rough estimation, it does nicely frame the significance of our model's results.

6 Conclusions

In this work, we propose a model to recommend the imaging protocols given doctors' requests as well as basic patient information. We developed a novel labelled dataset for this task and will look at releasing it pending we get permission from the Stanford hospital. We develop a model that achieves

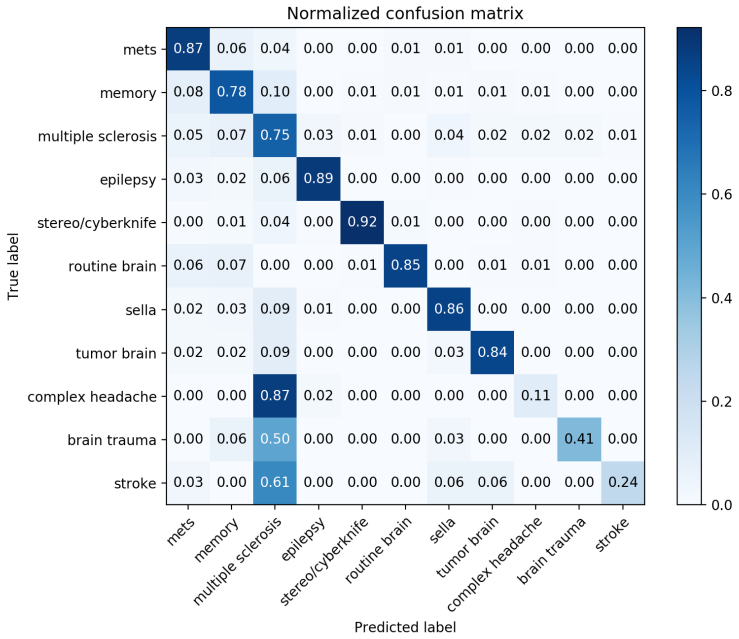

Figure 4: TongAI confusion matrix.

significantly better results than the baseline model and an overall high level of accuracy considering the level of error in the data. The model that achieved these results was relatively simple compared to model architectures that have become predominant in the text classification domain because this model drew most of its predictive accuracy from word features and efficient document representation techniques.

Our work was limited by the amount of data available and the data imbalances across classes. Although we had access to a very large dataset, only a small amount was usable. Future work would be mostly focused on creating a larger and more uniform dataset. This would require more time and support from medical professionals at the Stanford Hospital to be able to understand and parse through the larger dataset, which requires a greater understand of the protocol system at the hospital than we have. After having developed this larger dataset, we could move into classifying across the different regions, as well as making contrast recommendations.

Acknowledgments

This project was the idea of Doctor Elizabeth Tong, a neuroradiologist at the Stanford hospital. She saw the possibility of using deep learning for this task and also helped secure the data that we use in training and testing. We thank her of all the time and effort she put into making this project possible. Our project mentor is Pratyaksh Sharma.

References

[1] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics, 5:135–146.

[2] Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2016). Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759.

[3] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Adv. NIPS

[4]Wang Y, Liu S, Afzal N, Rastegar-Mojarad M, Wang L, Shen F, et al.(2018) A Comparison of Word Embeddings for the Biomedical Natural Language Processing.

[5]Dubois S, Romano N, Kale, D, Shah N, Jung K. (2018) Efficient Representations of Clinical Text

[6]Yoon Kim. (2014) Convolutional Neural Networks for Sentence Classification