Frailty Classifier Project
Modeling Strategy

Andrew Crane-Droesch

February 18, 2020

Outline

1. Brief overview of the project
2. Annotation: why/what/how
3. Word embeddings
4. Modeling
5. Active learning

Project overview

"Classify potentially preventable phenotypes of hospital admission among community-dwelling patients with chronic lung diseases using NLP within EHR data"

- Many hospital admissions are "low value"
  - Perhaps they could have been prevented
  - Patients often prefer not to be hospitalized
- It is not obvious *a priori* which people would be candidates for preventative measures, or what preventative measures to take
- This project builds on a related project that *identifies phenotypes* of people who are at high risk of hospitalization ("frailty")
- The **immediate aim** is to identify *actionable aspects* of frailty
- Later, the goal is to use this to identify patients at high risk of being hospitalized, who have ***actionable*** risk reduction avenues
  - Pulmonary rehab might prevent some respiratory crises
  - Physical therapy might enable pulmonary rehab
  - Installing grab bars in bathrooms might prevent falls

Project workflow

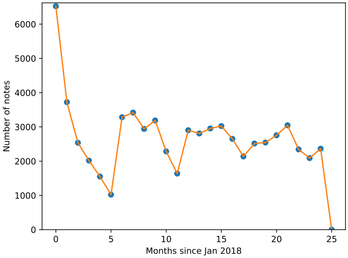
1. Download and process free text notes →
2. Annotation of spans of notes that are **positive** or **negative** for the following actionable aspects of frailty:
   - Fall risk
   - Musculo-skeletal problems
   - Respiratory impairment
   - Nutrition problems
3. Fitting models to predict the annotations from the spans
4. Using the fitted models to serve new batches of notes to annotate
5. Repeating 2, 3, and 4 until the models are good enough to classify text in the wild

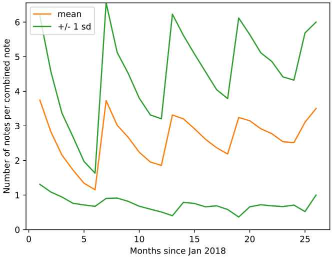
Data munging and text processing

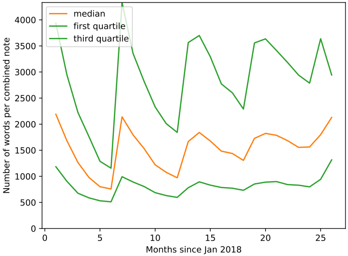
Population: all patients who've been at Penn for >1 year, with at least one of several diagnoses relating to chronic lung disease.
Pipeline:

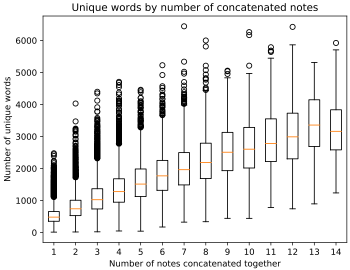
- Pull all signed **outpatient** progress notes from patients with relevant diagnoses
- Concatenate notes for individual patients into 6-month windows, simulating a prospective trial where targeting is done based on 6 months of history

Summary stats









Annotation

Annotation

1 -- [redacted] -- r

resp_imp_yes

2 [redacted] is_a 75 y.o. male. he_has a_history_of nodular_sclerosing hodgkin_lymphoma_with pulmonary_parenchymal involvement and sp

resp_imp_yes

LLL endobronchial obstruction/RML extrinsic compression now resolved, hypothyroidism, prior 20py smoking_history (quit in_his 40s),_a

3 he_was_diagnosed with lymphoma in_ [redacted] based_on supraclavicular_lymph_node biopsy and treated_with AVD with last treatm
showed_gradual_improvement with_complete metabolic_response on_june scan, followed_by increasing fdg_activity in abdominal_lymp
also_demonstrated new, numerous, small perilymphatic nodules concerning_for recurrent lymphoma vs sarcoid-like_reaction.

4 Subsequent bronchoscopy_confirmed sarcoid. he_did_not tolerate prednisone due_to irritability_and was hesitant_to try other_immunos
imaging at_this_time.

5 I last saw_him on_ [redacted] at_that_time, I recommended: - Stop bactrim, famotidine - PFT today - PFT, 6MWT at next_visit in 3 mo
develops new respiratory, cardiac, dermatologic, or_ocular symptoms - will_perform EKG at next_visit respiratory_symptoms_are stable

resp_imp_yes

6 Dyspnea after hot shower, prolonged moderate_exertion such_as walking outside home for_more_than 10 minutes.

7 Has an albuterol nebulizer at_home_but has_not tried using this to_see_if it alleviates symptoms.

8 lymphoma_is stable.

msk_prob_yes

9 Planning to_have hip_replacement in Fall as joint_pain often limits his activity.

msk_prob_yes

10 Will do PT afterwards.

Tokenization and label processing

```
>>> i+=20; x.iloc[i:(i+20)]
            token  length   end  start  Frailty_nos  Msk_prob  Nutrition  Resp_imp  Fall_risk
560             ,       2  3190   3188            0         0         -1         0          0
561          well       4  3194   3190            0         0         -1         0          0
562             -       1  3195   3194            0         0         -1         0          0
563     nourished       9  3204   3195            0         0         -1         0          0
564             ,       2  3206   3204            0         0          0         0          0
565           and       4  3210   3206            0         0          0         0          0
566            in       3  3213   3210            0         0          0         0          0
567            no       3  3216   3213            0         0          0         0          0
568      distress       8  3224   3216            0         0          0         0          0
569             .       2  3226   3224            0         0          0         0          0
570            No       3  3229   3226            0         0          0         0          0
571      distress       8  3237   3229            0         0          0         0          0
572             .       1  3238   3237            0         0          0         0          0
573            \n       2  3240   3238            0         0          0         0          0
574          HENT       4  3244   3240            0         0          0         0          0
575             :       1  3245   3244            0         0          0         0          0
576            \n       2  3247   3245            0         0          0         0          0
577          Head       4  3251   3247            0         0          0         0          0
578             :       2  3253   3251            0         0          0         0          0
579  Normocephalic     14  3267   3253            0         0          0         0          0
>>>
```

↑ this is the raw data. How can we use the token to predict the label?

Word embeddings

Document matrix

Start with a sparse matrix "one-hot" encoded representation of a document:

- The rows correspond to positions in a sequence
- The columns correspond to unique words
- The sum of the matrix equals the number of rows

$$
\boldsymbol{D} \equiv \begin{array}{c|cccccc}
 & I & am & Sam & would & you & like \\
\hline
I & 1 & 0 & 0 & 0 & 0 & 0 \\
am & 0 & 1 & 0 & 0 & 0 & 0 \\
Sam & 0 & 0 & 1 & 0 & 0 & 0 \\
Sam & 0 & 0 & 1 & 0 & 0 & 0 \\
I & 1 & 0 & 0 & 0 & 0 & 0 \\
am & 0 & 1 & 0 & 0 & 0 & 0 \\
would & 0 & 0 & 0 & 1 & 0 & 0 \\
you & 0 & 0 & 0 & 0 & 1 & 0 \\
like & 0 & 0 & 0 & 0 & 0 & 1
\end{array}
$$

Call it $\boldsymbol{D}$. It has dimensions $N \times V$ – total words by unique words.

Document matrix

- Document matrices are *sparse*. Lots of zeros.
- They are high-dimensional. Using $\boldsymbol{D}$ directly as a design matrix in a model is generally inefficient.
- A document matrix doesn't explicitly encode any information about how words might be similar to each other. Synonyms are wholly different from each other.
  - The euclidean distance between any two word vectors (rows in the matrix) will always be $\sqrt{2}$
  - The cosine similarity will always be zero
- Summing them column-wise makes a unigram matrix:

	I	am	Sam	$would$	you	$like$
$1-gram$	2	2	2	1	1	1

Embeddings

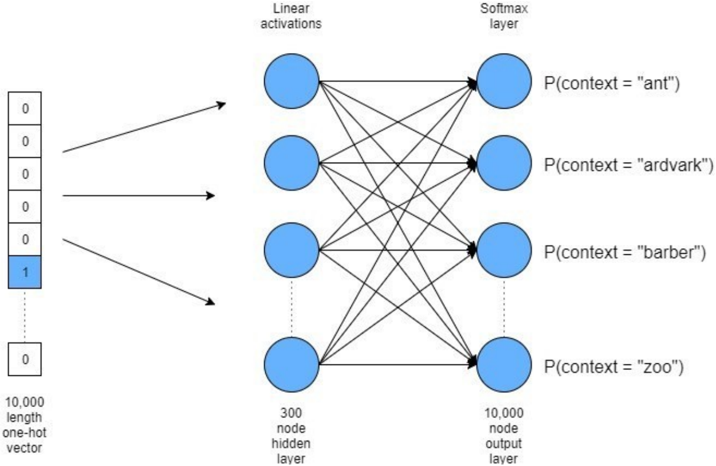
Embeddings attempt to do the following:

- Reduce the dimensionality of $\boldsymbol{D}$ without losing too much information
- Capture the similarity between similar words

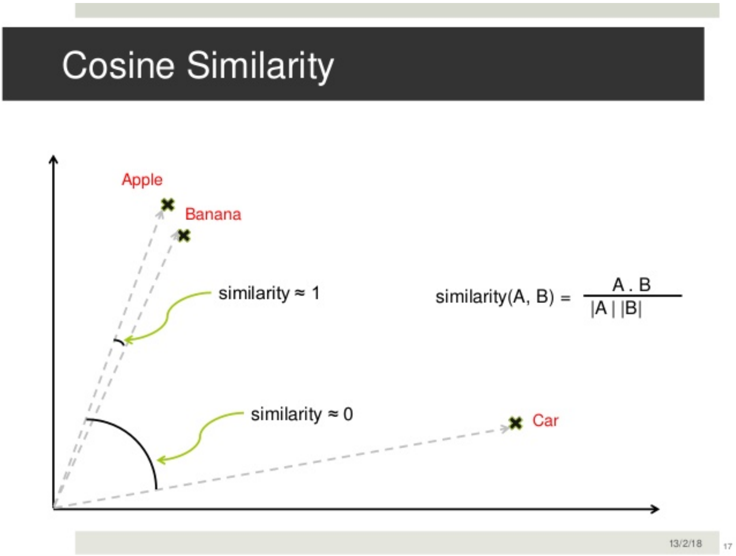
Different embedding methods do this in different ways, but all create analogous output:

- word2vec, fasttext, GloVe

Embeddings in pictures



Embeddings in pictures



Embeddings in math

(a simple) Embedding model:

$$\boldsymbol{D} = s\left(\boldsymbol{D}\boldsymbol{\Gamma}_0\boldsymbol{\Gamma}_1\right) + \epsilon$$

where:

- $\boldsymbol{D}$ is the document matrix, dimension $N \times V$
- $\boldsymbol{\Gamma}_0$ is the first weight matrix. It has dimension $V \times U$, where $U << V$. It is responsible for "embedding" $\boldsymbol{D}$ in a lower-dimensional space
- $\boldsymbol{\Gamma}_1$ is the second weight matrix. Its dimension is $U \times V$. It is responsible for taking the lower-dimensional representation and bringing it back to the original resolution.
- $s()$ is the softmax function. It maps the maps real numbers to a vector of probabilities summing to 1.
- ϵ is the error to be minimized. As the bottleneck gets skinnier (i.e.: as U gets smaller), ϵ will have to increase.

Embeddings in math

(a simple) Embedding model:

$$\boldsymbol{D} = s\left(\boldsymbol{D}\boldsymbol{\Gamma}_0\boldsymbol{\Gamma}_1\right) + \epsilon$$

where:

- The embedded text, call it $\boldsymbol{E}$, is defined as $\boldsymbol{D}\boldsymbol{\Gamma_0}$ (dimension $N \times U$)
- (most) Embedding models are trained by computing the derivatives of the parameter matrices $\boldsymbol{\Gamma}_0, \boldsymbol{\Gamma}_1$ with respect to a loss function, changing the parameters to make the loss smaller, and then recomputing the derivatives and repeating. This is backpropagation.
- the error ϵ is implicit. The goal is to minimize it, to get the best possible $\boldsymbol{\Gamma}$. But as the dimension of $\boldsymbol{\Gamma}$ gets smaller, this gets harder to do.

Embeddings in practice

Training and using embeddings are separate processes. Trained embedding models give you $\mathbf{\Gamma}_0$. You then supply your own text to multiply it against. For example, take $\boldsymbol{D}$ to be our Dr Seuss example from before. The embeddings $\mathbf{\Gamma}_0$ might look like this:

$$\mathbf{\Gamma}_0 \equiv \begin{array}{c|cc} & u1 & u2 \\ \hline I & 1 & 2 \\ am & 3 & 4 \\ Sam & 5 & 6 \\ would & 7 & 8 \\ you & 9 & 10 \\ like & 11 & 12 \end{array}$$

(Note that the entries of the matrix are totally unrealistic in this example)

Embeddings in practice

Say you wanted to embed the phrase “I like Sam” into this two-dimensional space:

$$
\begin{array}{r|cccccc} I & & & & & & \\ like & & & & & & \\ Sam & & & & & & \end{array}
\left[\begin{array}{cccccc} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{array}\right]
\left[\begin{array}{c|cc} & u1 & u2 \\ \hline I & 1 & 2 \\ am & 3 & 4 \\ Sam & 5 & 6 \\ would & 7 & 8 \\ you & 9 & 10 \\ like & 11 & 12 \end{array}\right]
=
\left[\begin{array}{c|cc} & u1 & u2 \\ \hline I & 1 & 2 \\ like & 11 & 12 \\ Sam & 5 & 6 \end{array}\right]
$$

Because the the document matrix $\boldsymbol{D}$ has only one “1” per row, this matrix multiplication is the same as a “lookup” operation.

Still, the matrix notation will come in handy later.

Word embeddings to sentence or span embeddings

Supervised learning jobs can all be conceptualized as

$$y = f(\boldsymbol{X}) + \epsilon$$

where y is some label and $\boldsymbol{X}$ is a matrix of features. So let's add a label to our toy example:

	I	*am*	*Sam*	*would*	*you*	*like*	**label**
I	1	0	0	0	0	0	0
am	0	1	0	0	0	0	0
Sam	0	0	1	0	0	0	1
Sam	0	0	1	0	0	0	0
I	1	0	0	0	0	0	0
am	0	1	0	0	0	0	0
would	0	0	0	1	0	0	1
you	0	0	0	0	1	0	1
like	0	0	0	0	0	1	0

(I labeled words that come after a verb)

Word embeddings to sentence or span embeddings

If we fit

$$\mathbf{label} = f(\boldsymbol{D}) + \epsilon$$

or

$$\mathbf{label} = f(\boldsymbol{E}) + \epsilon$$

our model would do nothing but capture the correlations between individual words and labels. But what if we want to capture the surrounding context?

First, let's remember that the matrix equation above is the same as

$$label_i = f(E_i) + \epsilon$$

where i indexes rows from 1 to N. A natural strategy could be to include lags and leads:

$$label_i = f(E_i, E_{i-1}, E_{i+1}) + \epsilon$$

But lots of other possibilities exist.

Word embeddings to sentence or span embeddings

Consider the span "I am Sam". Its embedding is:

	$u1$	$u2$	**label**
I	1	2	0
am	3	4	0
Sam	5	6	1

The centroid word is "am", which is associated with a label of zero. How do we associate information about the surrounding words with that zero label? Several options, all of which involve "windowing":

- Lags and leads
- span-wise maxes and mins, averages

	$u1$	$u2$	**label**	lag		$lead$		min		max	
am	3	4	0	1	2	5	6	1	2	5	6

In this toy example, the lags and leads happen to be the same as the maxes and mins, but that won't generally be the case.

Word embeddings to sentence or span embeddings

We've been working with a bandwidth of 1 (as measured from the centroid.). This is the same thing as a window size of 3.

Taking an average over this window is the same as doing the following:

$$[.33, .33, .33] \left[\begin{array}{c|cc} & u1 & u2 \\ \hline I & 1 & 2 \\ am & 3 & 4 \\ Sam & 5 & 6 \end{array} \right] = [3, 4]$$

But why weight evenly? We could do

$$[.2, .7, .1] \left[\begin{array}{c|cc} & u1 & u2 \\ \hline I & 1 & 2 \\ am & 3 & 4 \\ Sam & 5 & 6 \end{array} \right] = [2.8, 3.8]$$

Optimal weighting schemes could be hyperparameters. Or they could be *directly estimated* at the same time as model parameters if we're using a neural network. This is basically the same thing as a one-dimensional *convolution*.

Embedded Ngrams

Ngrams are usually computed over the whole of a document, but they could also be computed over a window. Here's bigrams for our dummy example:

$$\boldsymbol{G}^2 = \begin{array}{c|cccccc} & (I, I) & (I, am) & (I, sam) & \dots & (am, Sam) & (Sam, Sam) \\ \hline I & 0 & 1 & 0 & \dots & 0 & 0 \\ am & 0 & 1 & 0 & \dots & 1 & 0 \\ Sam & 0 & 0 & 0 & \dots & 1 & 0 \end{array}$$

These bigrams are going to be much too high-dimensional to use as-is, but we can embed them into a lower-dimensional space:

$$E^2 = \underbrace{\boldsymbol{G}^2}_{N \times V^2} \left[\underbrace{\boldsymbol{\Gamma}_0 \otimes \boldsymbol{\Gamma}_0}_{V^2 \times U^2} \right]$$

Doing so could help capture topology-dependent phenomena like negation and adjectives/adverbs. The dimensionality is too high to do this with the whole vocab even at $U = 100$, but this could be reduced through TF-IDF weighting of bigrams and PCA on the matrix $\boldsymbol{\Gamma}$.

Models

Given a label and a transformation that takes the document matrix $\boldsymbol{D}$ and returns $\boldsymbol{X}$, fitting a statistical model is generally a fairly straightforward exercise of tuning hyperparameters.

Neural nets are the exception, because they effectively create their own design matrices through representation learning. I'll discuss those later.

We'll try:

- Penalized logistic regression
- Random forest
- XGboost (with trees, probably)

Structured data (not incorporated yet)

We'll be using:

- Labs
- Comorbidities
- Frailty indices from the literature
- Vitals
- Demographics

All of the structure data will only vary at the *patient level*. We'll have $>> 10^6$ labeled tokens, but from $<< 1000$ labeled patients.

Random Forest

Doing the RF as a first pass, because RF's require minimal tuning.

Key considerations:

- The data aren't IID: X_i will be deeply correlated with X_{i+1}
  - because the features include window statistics
  - because the tokens represent language, which is non-random

  → Training and test sets need to be *cluster-sampled*
- Labels are rare, and classification forests aren't probablistically calibrated: their score isn't in general the expectation of a class's probability
  → The `ranger` package (in R) offers the *probability forest* option. Terminal nodes output probabilities, rather than majority votes. This will help in the early stages of active learning.

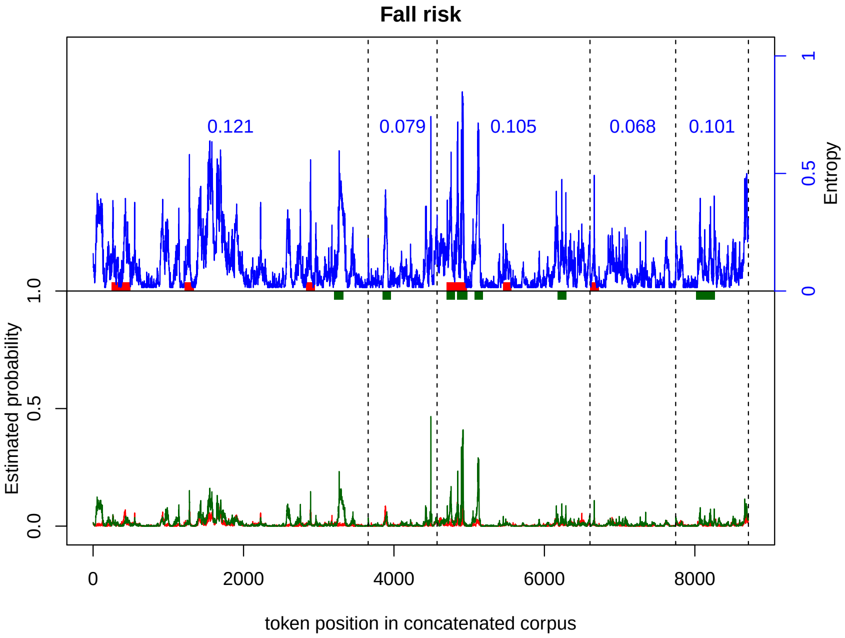
Featurization

Here are the sets of features of the embeddings that we're using:

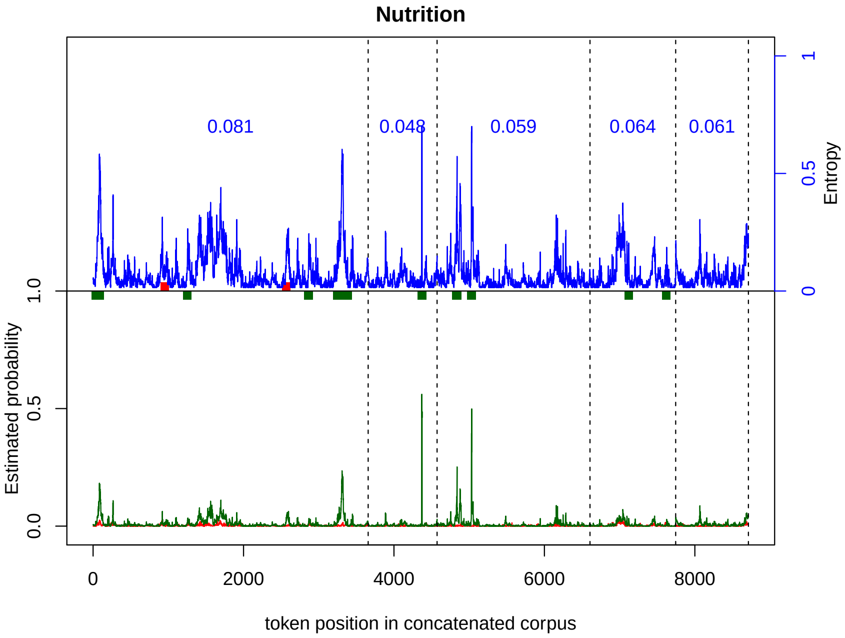
- Identity: $f(i) = E_i$
- Last word (one lag): $f(i) = E_{i-1}$
- Last word (two lags): $f(i) = E_{i-2}$
- Weighted mean over bandwidth (b): $f(i) = k\boldsymbol{E}_{i \in b}$
  - where k is a $1 \times b$ vector of interpolated densities from under a standard normal between -3 and 3
- Max over bandwidth: $f(i) =$ `lambda x: np.amax(x, axis=0)`
- Min over bandwidth: $f(i) =$ `lambda x: np.amin(x, axis=0)`

Bandwidth set to 30

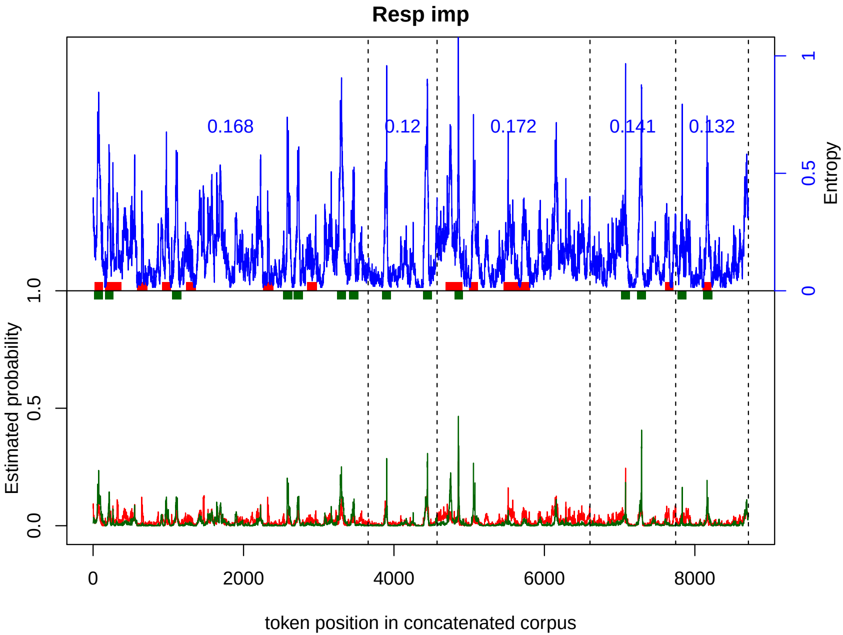
Results (preliminary)



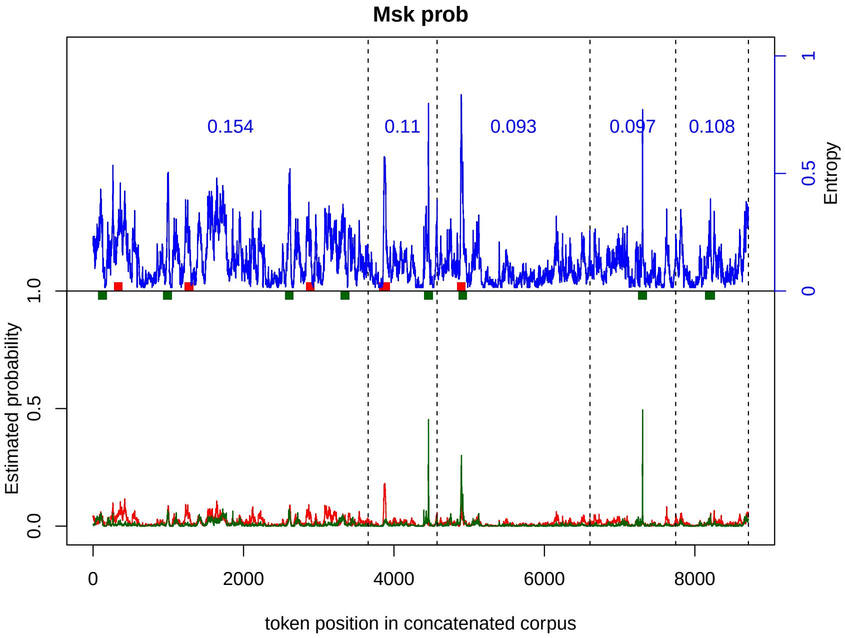
Results (preliminary)



Results (preliminary)



Results (preliminary)



Neural nets

- I haven't trained a neural net yet
- I expect that it will be much more efficient
- Neural nets have the nice feature that they implicitly do the featurization for you
  - That's what *representation learning* is, basically
- That doesn't mean that featurizations fed to the other models wouldn't be useful to the net
  - Basically it'd be giving the AI a cribsheet
- Likewise it'll be important to think carefully about model architecture, especially around questons of context (negation, allusion, etc.)

Active learning

How do we take the output of models fit to the first 25 patient notes and use them to pick the most fruitful notes to annotate next?

- Standard approach is to pick examples that we're maximally uncertain about
- *Entropy* is a useful measure of uncertainty from information theory

$$\text{Entropy} \equiv \mathrm{H} = -\sum_{c} p(class = c) log(p(class = c))$$

- It is maximized when you know nothing about which class an example belongs to.
- We're going to pick notes that have a lot of entropy.
- Not obvious how best to aggregate entropy over the note. Thinking perhaps an average over the top quantile. Maybe decile.

Active learning

Example: In the beginning, the model isn't very good and will default to the dominant class (neutral)

- A token that the model thinks is neutral will have low entropy: $\mathrm{H}([.01, .98, .01] = .11)$
- If the model thinks that *maybe* another token might be positive, entropy will be a little lower: $\mathrm{H}([.1, .88, .01] = .39)$. That token will make the note more likely to get picked.

Eventually, the model will get better about distinguishing the positive classes:

- After a while, it gets confident about a negative: $\mathrm{E}([.01, .01, .98] = .11)$, and that token won't influence what notes get picket to annotate
- If a note has examples that we are completely uninformed about (maximum entropy), we'd gravitate towards those notes: $\mathrm{H}([.33, .33, .33] = 1.1)$

Meta-considerations

This isn't a causal model, so the usual caveats about feature importance apply

- That said, it'd be useful to know what sort of features trigger positives and negatives for these frailty aspects
- Standard permutation importance is usually only done for RF's but can be coded for any model

Fairness:

- What happens if our model is much less certain about patients who are/aren't black/white/men/women/rich/indigent?
- Do we think it's likely that clinicians write differently about patients who are different – consciously or unconsciously, intentionally or otherwise?
- Can we apply post-hoc statistical fixes to correct for such biases?

Wrapup

Takeaways, challenges, and points to discuss:

- We are annotating and predicting labels on spans of text, but
  - how do we classify actual patients?
- Context is critical in text modeling.
  - What are the best ways to represent it?
- After the first batches of notes, we're going to pick subsequent notes about which we're maximally uncertain.
  - But what is the best way to aggregate that uncertainty?
- Bias is likely to pop up along the way.
  - How can we plan for it?

Thanks!