Sequence tagging for medical problems detection

Yacine Jernite

December 14, 2015

1 Introduction/Reminders

Our ultimate goal is to identify mentions corresponding to medical problems in text and map them to UMLS CUIs. Our initial two-steps approach was to use a simple CRF for mention detection (step 1), and focus on a more elaborate method to map mentions to CUIs, or even identify mistakes of the detection (step 2).

The reasoning behind this approach was that we felt we could make use of more interesting techniques for the second step, including leveraging MIMIC text for semi-supervised learning, and using the MRF model we developed for our ICML paper as a prior on the CUI distribution. To that end, I built a model predicting a CUI given a mention, its neighbouring words and neighbouring mentions. I first tried to do semi-supervised learning without using the MRF prior by training on MIMIC matches with different kinds of noise. I was unable to get any improvement from the MIMIC data.

While there are some things we could try to get this working, I'm very concerned that we won't get any results worth presenting even if we do. Indeed, the best F measure we are currently getting for the **detection** step is currently 0.76. To put that in perspective, the best team of the 2015 challenge got a **global** F measure of 0.757. The best **detection** accuracy of 2014 (numbers were not reported for 2015) is 0.813.

Between that and the fact that we will have to tackle step 1 more satisfactorily anyway if we want to release a tool which people can use, I grew increasingly uneasy with our approach, and finally decided to switch gears to try and improve the detection performance a few weeks ago. I'm describing the methods and current results here.

2 Model description

2.1 CRF

We were previously using a CRF with the following features (and combinations thereof):

- word
- lemma
- pos (part of speech)
- normal
- word_length
- prefix

- suffix
- all_caps
- capitalized
- word_pos (word position in the sentence)
- sentence_pos (sentence position in the document)
- sentence_length
- med_prefix (when the word prefix matches a UMLS prefix)
- umls_match_tag_full
- umls_match_tag_prefix
- umls_match_tag_acro

To predict where the mentions are, I approximate the marginal log-probability of a mention by the average marginal log-probability of its component tags (and the one to the right), and threshold on this marginal log-probability. So the probability of having a contiguous mention ranging from token 2 to token 4, for example, is:

$$\log(p(mention_{(2,3,4)})) = \frac{\log(p(B_2) + \log(p(I_3) + \log(p(I_4) + \log(p(B_5 \vee O_5))}{4}$$

The model was trained with the CRF++ package, and achieved an optimal F-measure on the development set of 0.767, for a marginal probability threshold of 0.75.

2.2 New tagging models

I spent the last month or so implementing two other tagging models in TensorFlow (I wanted to learn the language, and it's nice to have something that interfaces easily with Python). The models are a neural network which predicts each tag independently(-ish) given the whole sentence (SequNN), and a hybrid neural network / CRF system (NN-CRF, the CRF potentials are the output of a neural network).

Both have a first layer which embeds each token of the sentence into a dimension d space. The embedding of a token is the sum of the embeddings of its features. I used a restricted list of features, as it did not look like it affected the performance much:

- lemma
- prefix
- suffix
- pos
- umls_match_tag_full

So the embedding of the token *complaining:* {*lemma: complain, pos: VBG, prefix:com-, suffix, -ing, umls_match_tag_full: O*} is:

$$w_{complaining} = w_{complain} + w_{VBG} + w_{com-} + w_{-ing} + w_O$$

2.2.1 SequNN

I then tried adding various neural network architectures on top of this first layer, and learning the model with SGD.

Convolutional layer My current network actually consists of a single convolutional layer with window size 5, followed by a ReLU. To be perfectly clear, a convolutional window takes a window $(w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2})$, concatenates them into a dimension $5d$ vector, then applies a linear transformation to dimension d'. A ReLU is then applied, followed by a linear transformation from d' to the number of tags (5 currently), followed by a Soft-Max. (A picture is on its way)

Bi-directional LSTM I also implemented a bi-directional LSTM RNN. I couldn't get it to help however (it did worse than the ConvNet on its own, and putting the ConvNet on top of the RNN layer didn't help).

Prediction targets I then first replaced the marginals of output by the CRF in Section 2.1 by the probabilities output by the NN. However, I could not get past an F-measure of 0.7 with this system.

My next step was then to replace the tag prediction layer by a window prediction layer: instead of giving the probability of a tag, the last layer gives the probability of a tag and its left and right neighbour (125 possibilities). This got me back to the same performance as the CRF.

2.2.2 NN-CRF

Model I also trained a neural network to output the potentials of a CRF on the tags. The current architecture is basically the same as for the per-tag (per-window) prediction: embedding layer to window 5 convolution to ReLU to pair-wise potentials (5 x 5 matrices).

Learning The model is trained by SGD on the joint likelihood of the sequence of tags (inference is pretty fast), and back-propagating from the potentials to the NN parameters. The best F-measure obtained is slightly better than the vanilla CRF (0.772). I'm hoping to play a bit with the structure and learning parameters to improve that.

I also tried pseudo-likelihood learning of the model, but this performed terribly: the prediction of a tag given its neighbours is near perfect (0.997 on training, 0.993 on validation), but the best F-measure is around 0.4.

2.2.3 Other implementation details

I use the Adam optimizer with a learning rate of 0.02 for NN-CRF and 0.001 for SequNN (higher leads to gradient explosion), and gradient clipping (I'm not sure it's woking). I have L1 regularization on the lemma embeddings, and L2 regularization on all the embeddings. I need to add L2 regularization to the other embedding. No dropout for now.

3 Next steps

3.1 New tagging scheme

Adding "head" tags.

3.2 Combining methods

Start by taking the max of the marginals output by SequNN and NN-CRF, and move on to more complex methods.

3.3 Better representation

Add a language modelling objective to get better embeddings.