# Assignment 4
# CSE 447/547M: Natural Language Processing

University of Washington

Due: March 7, 2019 at 11:59pm

In this assignment, you will derive, design, and implement a neural structured perceptron for sequence labeling.

## 1 Deriving the Neural Structured Perceptron (25%)

**POS Tagging** Given a natural language sentence $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_n \rangle$, the task of POS tagging is to find the most likely sequence of POS tags $\hat{\boldsymbol{y}} = \langle y_1, y_2, \ldots, y_n \rangle$. In the following examples we also let $y_0 = \text{START}$ and $y_{n+1} = \text{STOP}$.

### 1.1 Decoding

When performing structured prediction for sequence labeling, we seek to find the sequence of tags which maximizes some global scoring function:

$$\hat{\boldsymbol{y}} = \operatorname*{argmax}_{\boldsymbol{y}} S(\boldsymbol{x}, \boldsymbol{y})$$

where $S$ maps a pair $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ to a real value. We assume that we can compute the global score by computing the sum of local scores:

$$S(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{n+1} s(\boldsymbol{x}, i, y_{i-1}, y_i).$$

### 1.2 Designing $s$

The structured perceptron defines the factorization of $S$ and the loss function with which the model is trained (hinge loss), but leaves $s$ completely up to us. What this means is that we get to define any model which takes $\langle \boldsymbol{x}, y_{i-1}, y_i \rangle$ and outputs a score in $\mathbb{R}$. This model can be as complicated or simple as we like.

One way of defining the local scoring function is as a sum of unary and binary scores, sometimes called "log-potentials."[1] It is useful to think of these log-potentials as similar to the emission and transition log-probabilities in an HMM. For each index $i$ in our sequence, we define

$$\phi_u(\boldsymbol{x}, i, y) \in \mathbb{R}$$

as the *unary* log-potential of the label $y \in \mathcal{L}$ for position $i$. We also define

$$\phi_b(y, y') \in \mathbb{R}$$

[1] Why "log"? Because there is an alternate universe where $S$ is a product of local scores (called potentials), rather than a sum. When we work with $S$ as a sum of local scores, we refer to summands as log-potentials.

as the *binary* log-potential between labels $y$ and $y'$.

We can then let

$$s(\boldsymbol{x}, i, y_{i-1}, y_i) = \phi_u(\boldsymbol{x}, i, y_i) + \phi_b(y_{i-1}, y_i).$$

In an HMM, we learn the emission and transition (log) probabilities through relative frequency estimation. In a neural structured perceptron, we learn $\phi_u$ and $\phi_b$ through backpropagation. Note also that if we eliminate $\phi_b$, then we have reverted back to a classifier that makes an independent decision for each token (i.e., it's not *structured*).

### 1.3 Deliverables

1. Let, for every possible value of $y_j$, $\heartsuit_j(y_j) = \max_{y_1,\ldots,y_{j-1}} \sum_{i=1}^{j} s(\boldsymbol{x}, i, y_{i-1}, y_i)$. Derive the recurrence.
2. Give an algorithm for calculating $\hat{\boldsymbol{y}}$ with $\heartsuit$ and $s$.
3. Does this algorithm look familiar? Give the time complexity. Be sure to carefully argue your answer.
4. Design a neural model to calculate $s$. Remember that the model needs to take in $\boldsymbol{x}$, $y_{i-1}$, and $y_i$ and output a score in $\mathbb{R}$. Think about how we can "neurally" define the unary and binary potentials. Be sure to accurately describe your model with equations (it's completely fine to use high-level abstractions e.g. *an RNN is a map* $f : \mathcal{X}^* \to \mathbb{R}^{h \times d}$).

## 2 Implementing the Neural Structured Perceptron in AllenNLP (50%)

Now that we have derived the neural structured perceptron, we will implement it in AllenNLP! Download the data and starter code[2] to get started.

### 2.1 Dataset

The data you will be using for this assignment is the Georgetown University Multilayer corpus (GUM) [Zeldes, 2017]. GUM is a collection of annotated web texts which include interviews, news stories, travel guides, how-to guides, academic writing, biographies, fiction, and forum discussions. These are the files:

- `a4-train.conllu`: data for training.
- `a4-dev.conllu`: data to select and tune your models.
- `a4-test.jsonl`: unlabeled test data in the JSON Lines format. Do not use the test data in any way to train your models.

Note that these files are in the CONLL-U format, which you also saw in A3; you should be able to reuse your `DatasetReader` from that assignment.

### 2.2 Deliverables

1. Set up an AllenNLP `DatasetReader` that can load GUM and pass it to your model. You should be able to use your dataset reader from A3. [3]
2. Implement an AllenNLP `Model` that implements the structured perceptron you defined in section 1. If you decide to change your model once you start implementing it, discuss why and accurately describe your new model with equations.
3. Create an AllenNLP configuration file that uses your `DatasetReader` and trains using your model.
4. Report the accuracy of your model on the training and development sets across the hyperparameter values you tested (at most 5).

[2] http://courses.cs.washington.edu/courses/cse447/19wi/assignments/A4.tgz

[3] If you want, you can use our implementation at [this will be available once A3 submission completely wraps up –deric].

5. Set up an AllenNLP `Predictor` that outputs a file containing the tweet and the model's predicted POS tags. The output should be in a tab-separated format with one instance per line. The first part of each line will be the tokens, each separated by a space. The second part will be the tags, each separated by a space. You should be able to reuse your predictor from A3. [4]

Register your `DatasetReader` as **pos_tagger_read**, your `Model` as **structured_perceptron_tagger**, your `Predictor` as **pos_tagger_pred**, and place them in a folder called `pos_tagger`. Provide your configuration in `pos-tagger.jsonnet`.

## 3 Kaggle (25%)

Create a Kaggle account [5] and report your accounts details [6]. Once you have an account, access the competition. [7]

In order to create Kaggle submissions, use the `kaggle.py` script included in the starter code. To create a Kaggle submission, run `./kaggle.py <test-preds.tsv> <out.csv>`. The first argument is the filepath of the file generated by your AllenNLP predictor. The second argument is the output filepath where the Kaggle submission file will be created. This will create a csv file with two columns, `Id` and `Category`. Each token in the test data will be assigned a unique ID and will be paired with its predicted label. This file can be submitted to Kaggle for evaluation.

Each day (in UTC[8]), you will be allowed three submssions. For each of these submissions, a public score will be displayed on the leaderboard. This public score is calculated using only a portion of the entire test set. The final leaderboard will be calculated using the other portion. Carefully consider the implications of this ranking scheme! Before the end of the competitions, please mark on Kaggle which submission you would like to be used for evaluation (on the private test data).

### 3.1 Deliverables

1. Beat the random baseline with your model. Describe any changes you needed to make and your tuning procedure.
2. Beat the basic baseline. Describe any changes you needed to make and your tuning procedure.

## 4 Bonus (up to +15%)

Extra credit will be awarded for particularly well-performing solutions submitted to Kaggle. If you have a solution that achieves high dev accuracy and does well on the public Kaggle leaderboard, describe why you think your model performs well and how you arrived at this model.

### Hints and Resources

- Although the perceptron may be familiar, it will likely be unhelpful to try to connect neural structured perceptrons to the linear perceptron for binary classification. This is because the modern neural structured perceptron has evolved greatly from the original perceptron and is much more like a conditional random field (requiring inference, in this case, calls to the Viterbi algorithm, during training) and a neural network (with nontrivial subgradients requiring backpropagation).

[4] If you want, you can use our implementation at [this will be available once A3 submission completely wraps up –deric].
[5] https://www.kaggle.com/
[6] https://goo.gl/forms/iZMcFzQINlRfmdV62
[7] https://www.kaggle.com/t/ad75b5032b8747beab080c79dcec8f94
[8] https://www.timeanddate.com/time/aboututc.html

- Chapter 7 of the textbook [Eisenstein, 2018] goes over sequence labeling and includes information on the Viterbi algorithm and structured prediction.
- By default, an AllenNLP `Predictor` expects input data in the JSON Lines format. This means we can output predictions with our model by running
  `allennlp predict <model.tar.gz> a4-data/a4-test.jsonl --predictor pos_tagger_pred --include-package pos_tagger --output-file <out.tsv>`.

### Submission Instructions

Submit a single gzipped tarfile (`A4.tgz`) on Canvas.

- **Code**: You will submit your config file and model prediction file for the best models, along with a folder containing your `DatasetReader`, `Model`, and `Predictor`. We assume that you always follow good coding practicies (commenting, structuring), and these factors are not central to your grade.
- **Report** (use the filename `A4.pdf` and include it in the tarfile): Your writeup should be no longer than two pages, in pdf (one-inch margins, reasonable font sizes, LaTeX-typeset).
- **Kaggle**: Make sure to make submissions online. To keep everything fair, the competition will close on March 10, 2019 at 11:59pm for everyone. If you have late days, you can use them to make submissions on Kaggle. Any submissions made after you have run out of late days will not be counted. Be sure to select a submission that was made on time according to how many late days you have remaining.

### References


Jacob Eisenstein. *Natural Language Prcoessing*. 2018. URL https://github.com/jacobeisenstein/gt-nlp-class/blob/master/notes/eisenstein-nlp-notes.pdf.

Amir Zeldes. The GUM corpus: creating multilayer resources in the classroom. *Language Resources and Evaluation*, 51(3):581–612, 2017.