L&S II: Assignment 2

Prof. Noah Smith

Due: Tuesday, October 3 (part hardcopy, in class, part electronic)

The goal of this assignment is to get you working with log-linear models without worrying about decoding. Like assignment 1, there is a dataset for training and a dataset for testing. Your performance will almost certainly be better if you reserve part of the training data for "practice" testing (we call this a "development" or "dev-test" dataset), so you can tune your training methods for good generalization.

The modeling task is to predict a person's **login name** on a UNIX system, given their **actual name**. Note that both are sequences of symbols in a finite alphabet Σ, which includes the letters, spaces, possibly the digits, and possibly punctuation.

Because I don't want you to worry about decoding yet, the task will not be to guess a person's login name given only their actual name. Instead, your model will need to assign scores to rank a number of candidates, only one of which is correct. The formal definition of the task, *for this assignment*, is as follows.

- You will be given pairs of the form $\langle x_i, y_i \rangle$ (for $i \in \{1, 2, ..., N\}$) where x_i is a person's name and y_i is the person's login name. (Format: one i per line, y_i followed by a tab followed by x_i (which may contain whitespace) followed by a newline.)
- Your model will assign probabilities of the form $p(y \mid x)$. At test time, you do not need to compute normalized probabilities, only scores (which may be in the log domain).
- Given an actual name x and a list of possible login names $\langle y_1, ..., y_M \rangle$, your model will compute a score for each y_i. This score should be equal to $\log p(y_i \mid x) + C$ for some (unspecified) constant C.

It is up to you how to build this model. You are encouraged to use log-linear models *of one kind or another*. Some questions to think about:

- What are the features of your model?
- If you were to estimate the parameters of your model to maximize over $\vec{\theta}$ the function $\prod_{i=1}^{N} p_{\vec{\theta}}(x_i, y_i)$ (maximum likelihood), you would have to compute expected feature values under the joint distribution $p_{\vec{\theta}}(X, Y)$—a sum over a very large set of outcomes. If you were to maximize $\prod_{i=1}^{N} p_{\vec{\theta}}(y_i \mid x_i)$ (maximum conditional likelihood), you would

have to compute expected feature values under $\tilde{p}(X) \cdot p_{\vec{\theta}}(Y \mid X)$; again summing over a very large set (all possible login names). How will you design your model to avoid this problem?

- How will you estimate the parameters of the model?
- How will you prevent overfitting?

Deliverables

1. Build a "ranking" program. For each testing example (a testing example includes an actual name x and a list of possible logins) made available (a few days before the assignment is due), your program will output the list of logins, ordered by decreasing scores, along with the scores.[1] For each testing example file named `foo`, create a file `foo.out` that is the output of your program given `foo` as input. Tar and gzip these files together, and email them to the instructor by lecture time on the deadline. You don't need to turn in your program—unless I doubt the validity of your output and request to see the source code.
2. A one-page write-up describing how you built your model (see the list of questions above and be sure to give answers to them). You are welcome to use existing tools of your own or tools made available by others, as long as you cite them in your write-up. The ranking program must be your own code.
3. Answer this question: How might I have made this task a lot easier for you? Hint: what do you know at test time that you didn't know at training time, and how might that information have changed your solution to the problem?

Hand in items 2 & 3 during the lecture on the deadline date.

[1] Format: the program should output x, followed by a newline, followed by lines giving "y_i (tab) $\log p_{\vec{\theta}}(y_i \mid x) + C$ (newline)" sorted in decreasing order by the second field.