

# COL 776: Assignment 1 (Part B)

**Due Date: 11:50 pm, Friday September 18, 2015. Total Points: 46 (+ 6 extra credits)**

## Notes:

- You should submit all your code as well as any graphs that you might plot (see below).
- Include a **single write-up (pdf) file** which includes a brief description for each question explaining what you did. Include any observations and/or plots required by the question in this single write-up file.
- You can use any programming language from the set C++, Java, Python, Matlab. If you would like to use any other language, please check with us before you start.
- Your code should have appropriate documentation for readability.
- You will be graded based on what you submit as well as your ability to explain your code.
- Refer to the [course website](#) for assignment submission instructions.
- This assignment is supposed to be done individually. You should carry out all the implementation by yourself.
- We plan to run Moss on the submissions. Any cheating will result in a penalty of **-10** points on your total course score (in addition to a 0 on the assignment). Stricter penalties (**including a fail grade**) may follow.

1. **Reading Assignment (10 points)** Read the paper on “[Conditional Random Fields](#)” posted on the website. Write a 1000 word summary of the paper in your own words. Your description should include the motivation of the paper, key ideas presented and the experimental findings (DO NOT just copy paste the abstract of the paper). You should understand the paper well enough to implement the model/algorithms presented. For the purpose of this assignment, **you can ignore Section 4** which is about learning the parameters.

2. **OCR character Recognition using Graphical Models (36 points + 6 Extra Credit)**  
**Note: This problem has been borrowed (with slight modifications) from the Graphical Models course offered by Andrew McCallum at UMass Amherst.**

In this problem you have to implement and experiment with a undirected graphical model for the optical character word recognition task. We will be studying computer vision task of recognizing words from images. We can recognize a word by recognizing the individual characters of the word. However recognizing a character is a difficult task. Further, when each character is recognized independent of its neighbors, it can often result in words that are not there in the English language. So, in this problem we will augment a simple OCR model with additional factors that capture some of our intuitions based on character co-occurrences and image similarities. The undirected graphical model for recognition of a given word is shown in Figure 1 below.

It consists of two types of variables:

- **Image Variables:** These are observed images that we need to predict the corresponding character of, and the number of these image variables for a word is the number of characters in the word. The value of these image variables is an observed image, represented by an integer id (**less than 1000**). For the description of the model, assume the id of the image at position  $i$  is represented by ***img(i)***.

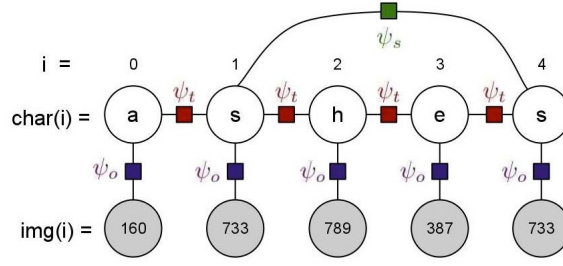


Figure 1: Undirected Graphical Model. Source: Graphical Models Course (Spring 2011) by Andrew McCallum, UMass Amherst.

- **Character Variables:** These are unobserved variables that represent the character prediction for each of the images, and there is one of these for each of the image variables. For our dataset, the domain of these variables is restricted to the ten most frequent characters in the English language (e,t,a,o,i,n,s,h,r,d), instead of the complete alphabet. For the discussion below, assume the predicted character at position  $i$  is represented by  $char(i)$ .

The model for a word  $w$  will consist of  $len(w)$  observed image ids, and the same number of unobserved character variables. For a given assignment to these character variables, the model score (i.e. the probability of the assignment according to the model) will be specified using three types of factors:

- **OCR Factors,  $\psi_o$ :** These factors capture the predictions of a character-based OCR system, and hence exist between every image variable and its corresponding character variable. The number of these factors of word  $w$  is  $len(w)$ . The value of factor between an image variable and the character variable at position  $i$  is dependent on  $img(i)$  and  $char(i)$ , and is stored in `ocr.dat` file described in the data section.
- **Transition Factors,  $\psi_t$ :** Since we also want to represent the co-occurrence frequencies of the characters in our model, we add these factors between all consecutive character variables. The number of these factors of word  $w$  is  $len(w) - 1$ . The value of factor between two character variables at positions  $i$  and  $i + 1$  is dependent on  $char(i)$  and  $char(i + 1)$ , and is high if  $char(i + 1)$  is frequently preceded by  $char(i)$  in English words. These values are given to you in `trans.dat` file described in the data section.
- **Skip Factors,  $\psi_s$ :** Another intuition that we would like to capture in our model is that similar images in a word always represent the same character. Thus our model score should be higher if it predicts the same characters for similar images. These factors exist between every pair of image variables that have the same id, i.e. this factor exist between all  $i, j, i \neq j$  such that  $img(i) == img(j)$ . The value of this factor depends on  $char(i)$  and  $char(j)$ , and is **5.0** if  $char(i) == char(j)$ , and **1.0** otherwise.

Given these factors, the probability of an assignment to the character variables of a word  $w$  according to our model will be given by:

$$P(chars|img) = \frac{1}{Z_{img}} \left( \prod_{\forall i} \psi_o(img(i), char(i)) \prod_{i=0, \dots, |w|-2} \psi_t(char(i), char(i+1)) \prod_{i,j | i < j, img(i) == img(j)} \psi_s(char(i), char(j)) \right)$$

where  $Z$  is the normalization constant as defined using sum over all possible assignments to character variables of the word. You can download all the data from [here](#).

The archive contains the following files:

- **Potential Directory:**

- **ocr.dat:** Contains the output predictions of a pre-existing OCR system for the set of thousand images. Each row contains three tab separated values "id a prob" and represents the OCR system's probability that image  $id$  represents character  $a$  ( $P(char = a | img = id) = prob$ ). Use these values directly as the value of the factor between image and character variables at position  $i$ ,  $\psi_o(image(i) = id, char(i) = a) = prob$ . Since there are 10 characters and 1000 images, the total number of rows in this file is 10,000.

- **trans.dat:** Stores the factor potentials for the transition factors. Each row contains three tab-separated values "a b value" that represents the value of factor when the previous character is "a" and the next character is "b", i.e.  $\psi_i(char(i) = a, char(i + 1) = b) = value$ . The number of rows in the file is 100 (10\*10).
- **Data Directory:**
  - **small/images.dat:** Contains observed images of one word on each row. The observed images for a word are represented by a sequence of tab-separated integer ids ("id1 id2 id3").
  - **small/words.dat:** Stores the true words for the observed set of images in the respective rows. True words are simply represented as strings (e.g. "eat"). You will need to iterate through both the files together to ensure you have the true word along with the observed images.
  - **large/allImagesX.dat:** Five files each containing image sequences for a larger set of words (can be thought of as a larger version of small/images.dat).
  - **large/allWords.dat:** Contains all the possible words which can be constructed from the given set of characters. Each row in this file stores the true word for the corresponding image sequence in allImagesX.dat file (can be thought of as a larger version of small/words.dat).

## Core Tasks

1. **Graphical Model (8 points):** Implement the graphical model containing the factors above. For any given assignment to the character variables, your model should be able to calculate the model score. Implementation should allow switching between three models:
  1. OCR model: only contains the OCR factors
  2. Transition model: contains OCR and Transition factors
  3. Combined model: containing all three types of factors

**Note:** To avoid errors arising from numerical issues, we suggest you represent the factors in the log-space and take sums as much as possible, calculating the log of the model score.
2. **Exhaustive Inference (8 points):** Using the graphical model, write code to perform exhaustive inference, i.e. your code should be able to calculate the probability of any assignment of the character and image variables. To calculate the normalization constant  $Z$  for the word  $w$ , you will need to go through all possible assignments to the character variables (there will be  $10^{len(w)}$  of these).
3. **Model Accuracy (8 points):** Run your model on the word images given in the file **small/images.dat**. For every image in the dataset, pick the assignment to character variables that has the highest probability according to the model, and treat this as the model prediction for the word. Using the true word value given in **small/words.dat**, compare the accuracy of the model predictions using the following three metrics:
  - i. Character-wise accuracy: Ratio of correctly predicted characters to total number of characters
  - ii. Word-wise accuracy: Ratio of correctly predicted words to total number of words
  - iii. Average Dataset log-likelihood: For each word given in **small/words.dat**, calculate the log of the probability of the true word according to the model. Compute the average of this value for the whole dataset.

Compare all of the three models described in (1) using these three metrics. Also give some examples of words that were incorrect by the OCR model but consequently fixed by the Transition model, and examples of words that were incorrect by the OCR, partially corrected by the Transition model, and then completely fixed by the Combined model.
4. **Running on the larger dataset (6 points):** Run your model on the larger version of the dataset (large/allImagesX.dat and large/allWords.dat). Report your findings. Note that is likely to take 15-20 times longer than the small dataset.

## Extra Credit (6 points)

**Varying the Potentials:** Since the only constraint on the factors is that they should be positive, one can imagine different values of these potentials can result in a higher accuracy. Is it possible for you to look at some of the errors and change the potentials to get a better accuracy? You should not change individual parameters given to you in the file, but instead of think of relative effect of the various types of factors. For example, what happens if you scale all the transition factor potentials by two, or square all the OCR potentials? Instead of directly "switching on" the skip factors, what if you plot the accuracy of the

model as you slowly increase the relative strength of the skip factors? Explore other such changes to the relative strengths of the factors, describe the thought process you used to change the potentials, its effect on accuracy and likelihood, and give examples improvements if any.