# Name: Chakrya Ros

# Section 2: Statistical Models (70 points)

1. For the following data, answer the given questions.

   a. What is the frequency of unknown words in the following training data if we choose a vocabulary before training? You may assume that your vocabulary includes <s> and </s> tokens.

   - **Close, excited, ski, ended, winter, ended, Swimming, close, excited, swimming, soon. So the frequency of unknown words are 11.**

   b. Give an example generated sentence where:
   - the sentence is not in the training data but the underlying words are - it could be generated using Shannon's method for generation

   **By using Shannon's method for the unigram, find the probability of frequency words and apply the random sentence according to the probability distribution with <s> as its first element. First split the words and add them into dictionary and count their frequency. Assign the frequency of that have valued equal to 1 as <UNK> and then calculate the probability of each word. In generating the sentence, I applied the Unigram that start with <s> and keep random selected the words if the word is not </s> token and update the start word to the current word. If the word is </s>, the generation sentence is done. Below is the result that generate two sentences.**

   **<s> season </s>**

   **<s> excited am <UNK> for </s>**

2. Using a Naive Bayes classifier, with bag-of-words features, what is P(y = spam | x) and what label will your classifier assign the sentence "remember to give Big Bird your password today !" given equal priors for each class and the following table of learned probabilities?

**P(y=spam | x) = P(y=sam)*P(x|y=spam) using Bayes theorem**

**For sentence X = "remember to give Big Bird your password today !", the word "Big, Bird, today and !" don't occur in the training set, so we drop them completely. After removing those words, computed as following:**

**P(y=Spam | x) = P(y=spam) * P(remember| y=spam) * P(to| y=spam) * P(give| y=spam) * P(your| y=spam) * P(password| y=spam)**

**P(y=Spam | x ) =( 0.5) * (0.27) * (0.01) * (0.16) * (0.20) * (0.11) = 0.00000475**

**P(y=not Spam | x ) =( 0.5) * (0.10) * (0.01) * (0.27) * (0.29) * (0.21) = 0.00000822**

**P(y=not Spam is greater than p(y=Spam) so the model predicts the class not spam for this sentence.**

3. What is the probability of "earthquake, no earthquake, earthquake" for the first three months of the year given the observation 10, 30, 10 insurance packages bought, and the following data?

**P(earthquake, no earthquake, earthquake, 10, 30,10) = P(earthquake | Start) * P(10 | earthquake) * P(no earthquake | earthquake) * P(30 | no earthquake) * P(earthquake | no earthquake) * P(10 | earthquake)**

**P(earthquake, no earthquake, earthquake, 10, 30,10) = 0.8 * 0.2 * 0.3 * 0.1 * 0.4 * 0.2**

**= 0.000384**

4. Is using the Viterbi algorithm equivalent to take the argmax of each column in a probability table? why/why not? Give an example sequence of observations that demonstrates your argument.

**No, the Viterbi algorithm does not take the argmax of each column in a probability table because the Viterbi algorithm moves from left to right iteratively to fill the columns in the table. Each column element contains the most probable path to reach this element and its probability. The first state of this algorithm fills the first column with the initial probabilities. The induction loop updates the values from t to t+1 by taking the maximum of all the incoming transitions for each element in the (t+1)th column and the node that led to it. Then, we find the most probable path from the maximum of all the elements of the last column in the table. Finally, we backtrack in the table to find the state sequence that led to it. I like to demonstrate my argument by using the probability of earthquake and not earthquake given the observation 10, 30, 10.**

**Fill the table below for running Viterbi algorithm given the same data as question 3:**

| | **10** | **30** | **10** |
|---|---|---|---|
| **earthquake** | **0.16** | **0.0448** | **0.0062** |
| **Not earthquake** | **0.1** | **0.006** | **0.0067** |

**The first column:**

- **P(earthquake | start)* P(10| earthquake) = 0.8*0.2 = 0.16**
- **P(no earthquake | start)* P(10| no earthquake) = 0.2*0.5 = 0.1**

**The second column:**

- **earthquake row:**
  - **0.16*p(earthquake | earthquake)* p( 30 | earthquake) = 0.16*0.7*0.4 = 0.0448**
  - **0.1*p(earthquake | no earthquake)* p( 30 | earthquake) = 0.1*0.4*0.4 = 0.016**
  - **Find the max(0.0448, 0.016) => 0.0448**
- **no earthquake row:**
  - **0.16*p(no earthquake | earthquake)* p( 30 | no earthquake) = 0.16*0.3*0.1 = 0.0048**
  - **0.1*p(no earthquake | no earthquake)* p( 30 | no earthquake) = 0.1*0.6*0.1 = 0.006**
  - **Find the max(0.0048, 0.006) => 0.006**

**The third column:**

- **earthquake row:**
  - **0.0448*p(earthquake | earthquake)* p( 10 | earthquake) = 0.0448*0.7*0.2 = 0.0062**
  - **0.006*p(earthquake | no earthquake)* p( 10 | earthquake) = 0.006*0.4*0.2 = 0.00048**
  - **Find the max (0.0062, 0.00048) => 0.006272**
- **no earthquake row:**
  - **0.0448*p(no earthquake | earthquake)* p( 10 | no earthquake) = 0.0448*0.3*0.5 = 0.0067**
  - **0.006*p(no earthquake | no earthquake)* p( 10 | no earthquake) = 0.006*0.6*0.5= = 0.0018**
  - **Find the max (0.0067, 0.0018) => 0.0067**

5. Why is a logistic regression classifier considered a discriminative model? (2 – 3 sentences)

**A logistic regression classifier is considered a discriminative model because it can learn which features are most useful and we compute posterior P(y|x) directly. Logistic regression also can be used to classify the observation into one of two classes (like "positive sentiment" or negative sentiment") or into one of many classes. When we used to solve NLP tasks, it estimates p(y|x) by extracting features from input text and multiplying each feature by the weight and adding them up and applying sigmoid function to get boundary between 0 or 1. For example, we want to classify the picture is dog (label "0") or cat (label "1").**

**By training the data that have features given labels, in the testing set, the output will classify the features that close to 0-1 boundary by using sigmoid function.**

6. Can we use the same algorithm to train an MEMM as we used for Logistic Regression? Why/why not? (2 – 3 sentences and/or example code)

**Yes, we can use the same algorithm to train an MEMM as we used for Logistic Regression because they are a discriminative model that we want to find the weights for each feature, that is, the weights that will make the training examples fit best the classes to which they belong. The MEMM is a multinomial logistic regression for classification. Given a sequence of observation, feature function and corresponding hidden states, we use stochastics gradient descent to train the weight to maximum the log-likelihood of the training corpus. MEMM train a binary logistic regression for each class by apply SoftMax function.**

7. In an HMM, we directly model transition probabilities between states at time *i* and *i-1*. Is this transition probability modeled in an MEMM? If yes, how? If no, why not?

**No, an MEMM does not capture the transitional probability between states at time *i* and *i-1.* The MEMM captures the conditional probability of the current state given the observation and the previous state $P(tag_i \mid word_i, tag_{i-1})$.**

8. You are designing a logistic regression classifier to assign movie reviews to the classes.

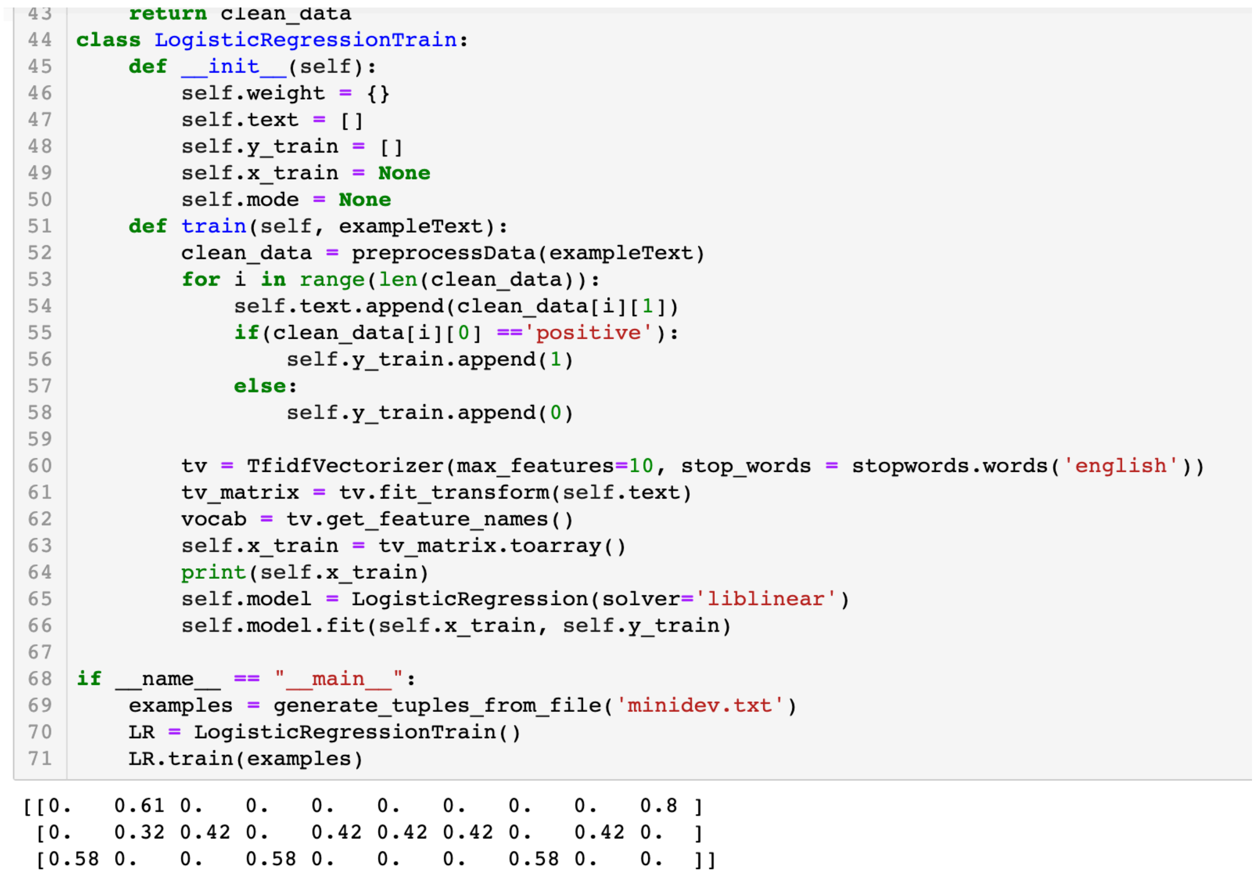
a. Give 4 different features:
   - **$x_1$ is represented the word count of document**
   - **$x_2$ is represented counting of positive lexicon in document**
   - **$x_3$ is represented 1 if 'no' or 'not' in document else 0.**
   - **$x_4$ is represented counting of negative lexicon in document**

b. Convert the given training data into the features:

   - **$x_1$ = 86 words in documents, take log(86) = 6.42**
   - **$x_2$ = [joy, brighter, foul-mouthed, funny, comedies, real]**
   - **$x_3$= 0**
   - **$x_4$ = [burst, little, loathing, mediocrity, nonexistence]**

**so we have features: [6.42, 7, 0, 5]**

**Below is the screenshot that I wrote in python to convert training data into feature vector:**



```
        return clean_data
class LogisticRegressionTrain:
    def __init__(self):
        self.weight = {}
        self.text = []
        self.y_train = []
        self.x_train = None
        self.mode = None
    def train(self, exampleText):
        clean_data = preprocessData(exampleText)
        for i in range(len(clean_data)):
            self.text.append(clean_data[i][1])
            if(clean_data[i][0] =='positive'):
                self.y_train.append(1)
            else:
                self.y_train.append(0)

        tv = TfidfVectorizer(max_features=10, stop_words = stopwords.words('english'))
        tv_matrix = tv.fit_transform(self.text)
        vocab = tv.get_feature_names()
        self.x_train = tv_matrix.toarray()
        print(self.x_train)
        self.model = LogisticRegression(solver='liblinear')
        self.model.fit(self.x_train, self.y_train)

if __name__ == "__main__":
    examples = generate_tuples_from_file('minidev.txt')
    LR = LogisticRegressionTrain()
    LR.train(examples)
```

```
[[0.   0.61 0.   0.   0.   0.   0.   0.   0.   0.8 ]
 [0.   0.32 0.42 0.   0.42 0.42 0.42 0.   0.42 0.  ]
 [0.58 0.   0.   0.58 0.   0.   0.   0.58 0.   0.  ]]
```

c. Report the assigned label from your mini classification:

**First review:**

- **x = [5, 1, 1, 3]**
- **z = [(5*0.5) + (2*1) + (1*-0.5) + (4*-1)] + 0.25 = -1.655**
- **sigmoid(z) = 0.56 => Positive**

**Second review:**

- **x = [4.17, 0, 0, 4]**
- **z = [(4.17*0.5) + (0*1) + (0*-0.5) + (4*-1)] + 0.25**
- **sigmoid(z) = 0.16 => Negative**

## Multiple Choice

9. In a training set with examples **X** and corresponding labels **Y**, a generative model would seek to learn ________ from the training data: (Choose all that apply)

**b. P(X|Y) d. P(Y)**

10. Which of the following shows a distribution that a generative model might learn? (Choose one)

**B.**

11.Select all of the ways in which we can deal with the unknown word problem during the training step of a language model: (Choose all that apply)

1. **Convert all low count words to <UNK> during training**
2. **Choose a vocabulary, convert all words not in vocabulary to <UNK>**
3. Convert all words with high character counts to <UNK> during training
4. Randomly convert .05 of the words in the training set to <UNK>
5. Nothing special needs to be done - language models trained on enough data are robust

   enough to not encounter problems with unknown words

12.In a language model, we want to _______ (maximize/**minimize**) perplexity and _______ (**maximize**/minimize) dev/test set probability.

13.What is it called when the weights for logistic regression features fit the training set perfectly? (Choose one)

**c. Overfitting**

14.Does an MEMM for POS tagging generate multinomial or binary classification?

**a. Multinomial**

15.Must an MEMM use a sigmoid or SoftMax classification?

**b. MEMMs must use the SoftMax function**

**Citations:**

- Quiz8
- Chapter 5 in textbook, https://web.stanford.edu/~jurafsky/slp3/5.pdf
- Chapter 4 in textbook, https://web.stanford.edu/~jurafsky/slp3/4.pdf
- https://en.wikipedia.org/wiki/Discriminative_model
- http://www.davidsbatista.net/blog/2017/11/12/Maximum_Entropy_Markov_Model/
-