CS7347 Test 1, Noah Gardner, 000843905

1. [**Points 5**] Which of the following are correct for the given regular expression `[\w.-]+@[\w.-]+`.

   a. `al.ice@yahoo.com`

   b. `bob.alice-com`

   c. `1.Abc@kennesaw.edu1`

   d. `@ksu.edu`

   e. `All of the above`

   Answer:

   The grouping `[\w.-]+` matches the any character in `[a-zA-Z0-9_.-]` between 1 and infinite number of times. The `@` matches the `@` symbol literally. b is missing the symbol `@`, and d misses the first grouping match. e is incorrect since b and d are incorrect. However, both a and c should match.

   Therefore, the answer is a and c.

2. [**Points 5**] Which of the followings are correct for the given regular expression `^[^A-Z0-6]+`.

   a. `^abc 10`

   b. `abc9`

   c. `1 "Hello"`

   d. `^abc9`

   e. `None of the above`

   Answer:

   The the first `^` token looks for a match at the front of the line. The second `^` token tells us to match any character not in `[A-Z0-6]` between 1 and infinite number of times. a is incorrect since it has a space in the middle, however, the first half would match. b will match since `abc9` are not in `[A-Z0-6]`. c is incorrect due to the space, also the first half of the line does not match. Finally, d will match since `^abc9` are not in `[A-Z0-6]`. e is incorrect since b and d match.

   Therefore, the answer is b and d.

3. [**Points 5**] Write down the differences of different activation functions, sigmoid, tanh, relu.

Answer:

Activation functions are used in neural networks and define the output of a node given it's inputs.

sigmoid

The sigmoid function is a function that outputs a value between 0 and 1. It is a non-linear activation function and is widely used due to differentiablity of the functon. However, the sigmoid function may cause the neural network to get stuck in local minima due to vanish gradient.

$$y = 1/1 + e^{-x}$$

tanh

The tanh function is a function that outputs a value between -1 and 1. It is a non-linear activation function similar to the sigmoid function. Due to the larger range of the tanh activation function, it can provide stronger gradients than the sigmoid function.

$$y = (e^{x} - e^{-x})/(e^{x} + e^{-x})$$

relu

The relu function is a function that outputs a value between 0 and ∞. It is a linear activation function, different from the sigmoid and tanh functions. Additionally, the relu function does not have the vanishing gradient problem, but instead has an opposite problem called the *exploding gradient* problem. The exploding gradient problem can also be present in other activation functions like tanh.

$$y = max(0, x)$$

4. [**Points 5**] What is sequence labelling? How would you build your parts of speech baseline model?

   Answer:

   Sequence labelling is a task in which a sequence of words is labeled with a sequence of parts of speech. Alternatively, sequence labelling can be used for name entity recognition. To build a baseline model for parts of speech tagging, you would require a training set of tagged sentences as well as a model that can take sequential data such as a sentence. Recurrent neural networks (RNN) work well for this task.

5. [**Points 10**] What are name entities? What is entity recognition task? Give an NER example. Why is Name Entity Recognition (NER) a difficult task? Please describe with example, how window based NER classification can be performed to detect Person Name.

   Answer:

   Name entities are entities that are recognized by a computer program, similar to proper nouns. Name entities are usually recognized by a computer program as a person, location, organization, or other entity. NER is difficult task because whether a word is a proper noun or not requires a lot of context that typically can't be provided by just the input sentence. It can also be hard to tell when the name entity starts and ends. Therfore, a lot of training data is required.

   Window-based NER can be used to help provide context for the word to be classified. For a given input sentence and a given word X_i we can train a softmax classifier. For example, for a window size of 3, we have the following input:

   $$X = [X_{i-2} * X_{i-1} * X_i * X_{i+1} * X_{i+2}]$$

   To predict whether the input X_i is a Person Name, we can use a low value for *not a Person Name* and a high value for *is a Person Name*. Then, given input X and output ground-truth labels, we can use backpropagation to train the model and make predictions.

6. [**Points 10**] Given the following equations:

   (a) $a = 3x + 4y$

   (b) $b = 2x - 7y$

   (c) $L = 2(a + b - z)$

   (a) Please draw computation graph (circuit diagram) for the given equation above. [**Points 2**]

   (b) Show forward pass values on the diagram, for the given values of x=3, y=- 2 and z=1. [**Points 3**]

   Answer:

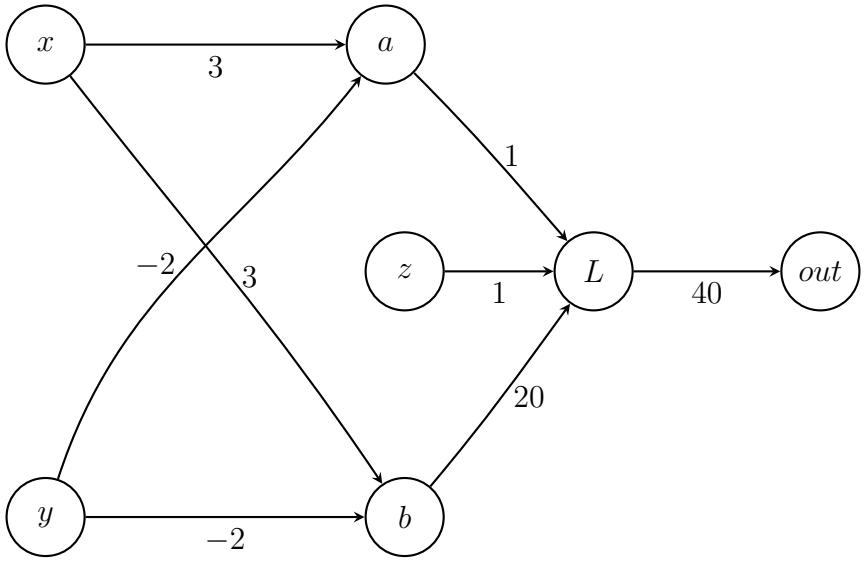


Figure 1: Computation Graph - Forward Pass

(c) Show a complete backpropagation circuit diagram with corresponding gradient values. **[Points 5]**

Answer:

The nodes x_2, y_2, and *out* are used to show the backprop to the required nodes due to limitations of the modelling software.

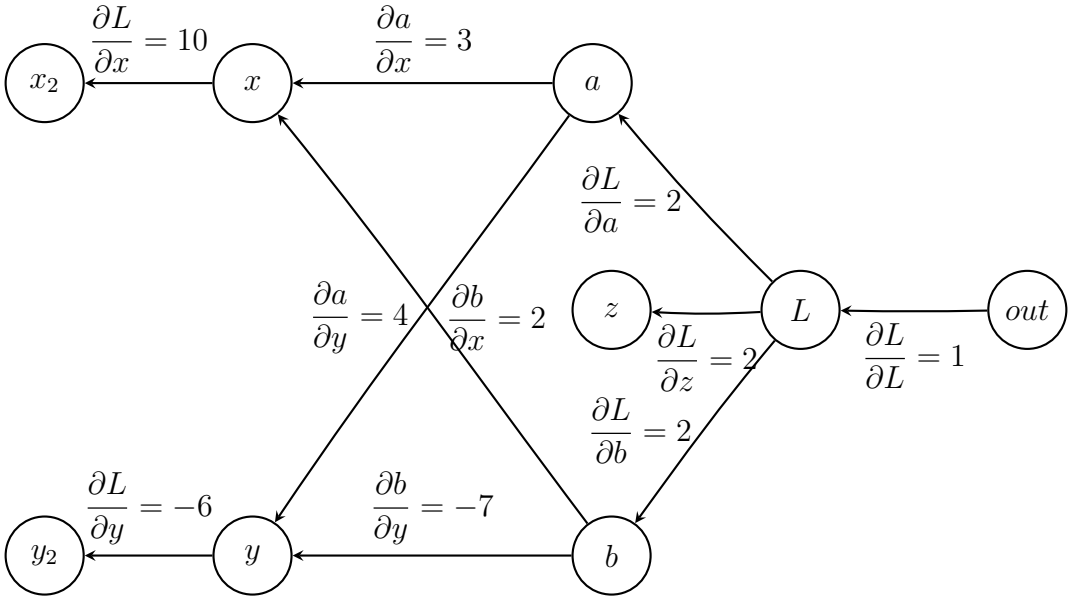


Figure 2: Computation Graph - Backpropagation

$\frac{\partial L}{\partial L} = 1$	$\frac{\partial a}{\partial x} = 3$
$\frac{\partial L}{\partial a} = 2$	$\frac{\partial a}{\partial y} = 4$
$\frac{\partial L}{\partial b} = 2$	$\frac{\partial b}{\partial x} = 2$
$\frac{\partial L}{\partial z} = 2$	$\frac{\partial b}{\partial y} = -7$

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial a}\frac{\partial a}{\partial x} + \frac{\partial L}{\partial b}\frac{\partial b}{\partial x} = 2 * 3 + 2 * 2 = 10$$
$$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial a}\frac{\partial a}{\partial y} + \frac{\partial L}{\partial b}\frac{\partial b}{\partial y} = 2 * 4 + 2 * -7 = -6$$

Figure 3: Partial Derivatives Calculations for Backpropagation

7. [**Points 20**] You are given a large text corpus and a word2vec embeddings. You are building your Neural Language model (NN-language model) for your given corpus with two hidden layers. Please assume that you can choose your own configuration for these two layers (i.e., number of neurons, etc.)

   a. Please mention how would you build your NN-language model using given word2vec embedding. Please show detailed diagram and language model equations (forward pass only) for each layer. [**Points 10**]
   Answer:

   Given word2vec embeddings, we can create a neural network language model. This model will consist of an input layer, which, on each timestep, will receive the word embedding of context words (previous words in the sentence). The next layer in the network is at least one hidden layer, although multiple can be used.

   The nodes in the input layer are connected to the nodes in the output layer with a weight matrix. The nodes in the hidden layer apply the activation function on the inputs and output the results to the output layer. In the case of a neural network language model, the output layer is a softmax layer.

   For our configuration, our NN-language model will consist of an input layer, a hidden layer, and an output layer. The input layer has a number of nodes equal to the size of the input window for context words. We will use a window size of 3. For simplicity, the hidden layer will have 2 nodes. Finally, the output layer has a number of nodes equivalent to the number of words in the vocabulary. Finally, our nodes will have a sigmoid activation function. **See Figure 9 in the Appendix for more details.**

   1. **First Layer (Input)**
      Given input context words, output word embeddings of context words from given embeddings. This will depend on the training dataset but takes the form of a vector. We apply a one-hot encoded vector to get our input values.
   2. **Second Layer (Hidden)**
      Word embeddings of context words from first layer multiplied by the weight W_{ij} from input node a_i to hidden node h_j plus the bias term b_j as input to the activation function.

      $$h_j = \sigma(W_{ij}a_i + b_j)$$

   3. **Third Layer (Softmax)**
      y_k is the probability that the next word is the word k. The output layer is a softmax layer, so the ouput from the previous hidden layer is passed through a softmax function to get the most probable word.

      $$\hat{y_k} = softmax(h)$$

b. You decided to use your own word embedding instead of word2vec. How would you design your NN-language model? Please show details diagram and forward pass equation for your language model. What are the parameters you need to train? [**Points 10**]]

Answer:

Since we will use our own word embedding, the first step is to describe how it will work. Word embeddings allow us to represent words in a vector space. We can use a frequency-based approach to create our word embeddings. We first count the number of occurences of each word in the training set. Then, we can normalize the word embeddings by using a standard normalization function, such as min-max scaling.

After creating these embeddings, we can use the same NN-language model as before. We have an input layer that takes the word embeddings and feeds into a hidden layer. Then a hidden layer feeds into our softmax output layer. Our proposed embeddings may be less robust than word2vec embeddings, however, they are simple to calculate. Other than frequencies, there are no new parameters to train compared to the previous model.

8. **[Points 15]** Please build your character-gram (char-gram) language models for the given training set. Please assume that you experiment will only have the following characters – [a, b, c, d, f, h] that exists in the training set.

   Training set:
   a a b c d b f
   b c h a d f f a h
   b b a a h c c h d d f
   a b f f c c d f h
   h h f f c a c d d d

   Test set:
   a b c d f
   a b b c c d

   Task:

   i. Build char-unigram language model. **[Points 3]**
   Answer:

p(a)	8/46
p(b)	6/46
p(c)	8/46
p(d)	8/46
p(f)	9/46
p(h)	7/46

Figure 4: Char-unigram Model

ii. Build char-bigram language model. [**Points 3**]
Answer:

p(a|a)	2/8
p(a|b)	1/6
p(a|begin)	2/5
p(a|c)	1/8
p(a|f)	1/9
p(a|h)	1/7
p(b|a)	2/8
p(b|b)	1/6
p(b|begin)	2/5
p(b|d)	1/8
p(c|a)	1/8
p(c|b)	2/6
p(c|c)	2/8
p(c|f)	2/9
p(c|h)	1/7
p(d|a)	1/8
p(d|c)	3/8
p(d|d)	3/8
p(d|h)	1/7
p(end|d)	1/8
p(end|f)	2/9
p(end|h)	2/7
p(f|b)	2/6
p(f|d)	3/8
p(f|f)	3/9
p(f|h)	1/7
p(h|a)	2/8
p(h|begin)	1/5
p(h|c)	2/8
p(h|f)	1/9
p(h|h)	1/7

Figure 5: Char-bigram Model

iii. Compute joint probability for the given test set using char-unigram model. [**Points 3**]
Answer:
Unigram Joint Probabilities:

a) `p(a b c d f)` =
$p(a) * p(b) * p(c) * p(d) * p(f) =$
$8/46 * 6/46 * 8/46 * 8/46 * 9/46 = 0.000134$

Therefore, the probability of the first sentence is **0.000134**.

b) `p(a b b c c d)` =
$p(a) * p(b) * p(b) * p(c) * p(c) * p(d) =$
$8/46 * 6/46 * 6/46 * 8/46 * 8/46 * 8/46 = 0.000015$

Therefore, the probability of the second sentence is **0.000015**.

iv. Compute perplexity of your models (char-unigram, char-bigram) and compare which model is better. [**Points 6**]
Answer:

Perplexity is defined as the inverse probability of the test sentence normalized by the number of words (characaters, in this case) in the sentence. The bigram model will have the tokens *begin* and *end* inserted at the beginning and end of each sentence to the probability of each character as the beginning and end of the sentence. These tokens will not count towards the total amount of words in the sentence during the normalization step.

Since we have the joint probability using the unigram model, calculate the probability of each sentence in the test set with the bigram model:

Bigram Joint Probabilities:

a) `p(a b c d f)` =
$p(a|begin) * p(b|a) * p(c|b) * p(d|c) * p(f|d) * p(end|f) =$
$2/5 * 2/8 * 2/6 * 3/8 * 3/8 * 2/9 = 1/960 = 0.00104$

Therefore, the probability of the first sentence is **0.00104**.

b) `p(a b b c c d)` =
$p(a|begin) * p(b|a) * p(b|b) * p(c|b) * p(c|c) * p(d|c) * p(end|d) =$
$2/5 * 2/8 * 1/6 * 2/6 * 2/8 * 3/8 * 1/8 = 1/15360 = 0.000065$

Therefore, the probability of the second sentence is **0.000065**.

Unigram Perplexities:

a) $(0.000134)^{-1/5} = 5.95085$

b) $(0.000015)^{-1/6} = 6.36773$

Average: $(5.95085 + 6.36773)/2 =$ **6.15929**

Bigram Perplexities:

a) $(0.00104)^{-1/5} = 3.94997$

b) $(0.000065)^{-1/6} = 4.9871$

Average: $(3.94997 + 4.9871)/2 =$ **4.46885**

The average perplexity of the bigram model is lower than the unigram model. Because a lower perplexity indicates a better performance, the bigram model is better than the unigram model.

9. **[Points 15]** Assume that we are in an alien world and their languages are different and only contains the following vocabulary [*delta, gamma, alpha, beta, sigma, derivative, summation*]. Their parts-of-speech tags are given as [**A, B, C, D**].

   You are given a task to assign tags using Hidden Markov Model for the given test sentence. Given the following sentences as training examples:

Sentence 1:	`delta gamma sigma summation`
Tags:	[**A, B, C, A**]
Sentence 2:	`alpha sigma beta derivative`
Tags:	[**A, C, D, A**]
Sentence 3:	`derivative gamma delta beta`
Tags:	[**A, B, B, D**]
Sentence 4:	`sigma summation beta alpha`
Tags:	[**C, B, C, D**]
Sentence 5:	`alpha beta sigma derivative`
Tags:	[**A, B, C, A**]

   Test Sentence: `alpha gamma beta sigma`

   a. Please calculate transition probabilities. **[Points 3]**
   Answer:

   To calculate transition probabilities, count the number of co-occurences for each tag and divide by the total number of times the tag appears. We add tags S and E to mark the beginning and end of the sentence.

Tag	A	B	C	D	E	totals
S	4/5	1/5	0	0	0	5
A	0	3/7	1/7	0	3/7	7
B	0	1/5	3/5	1/5	0	5
C	2/5	1/5	0	2/5	0	5
D	1/3	0	0	0	2/3	3

Figure 6: Transition Probabilities

b. Calculate emission probabilities. [**Points 3**]

Answer:

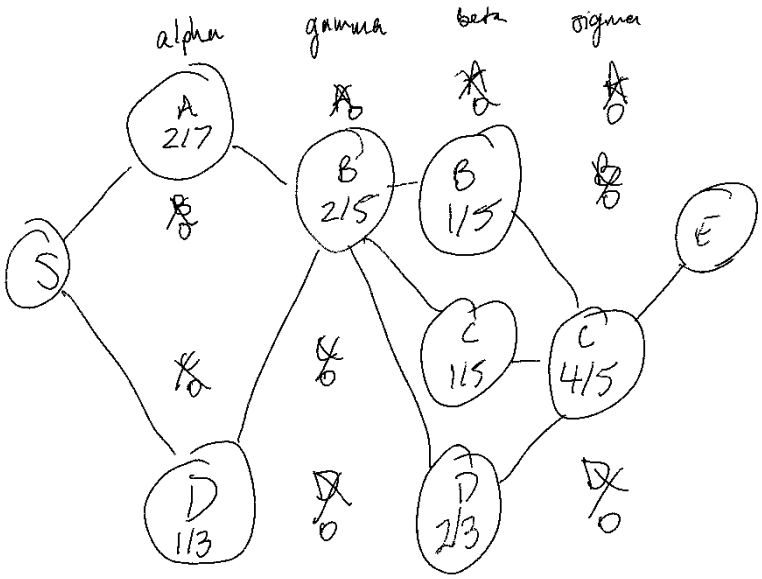
To calculate emission probabilities, count each time a word is associated with each label, divided by the total number of times a word is associated with that label.

Word	A	B	C	D
delta	1/7	1/5	0	0
gamma	0	2/5	0	0
alpha	2/7	0	0	1/3
beta	0	1/5	1/5	2/3
sigma	0	0	4/5	0
derivative	3/7	0	0	0
summation	1/7	1/5	0	0
totals	7	5	5	3

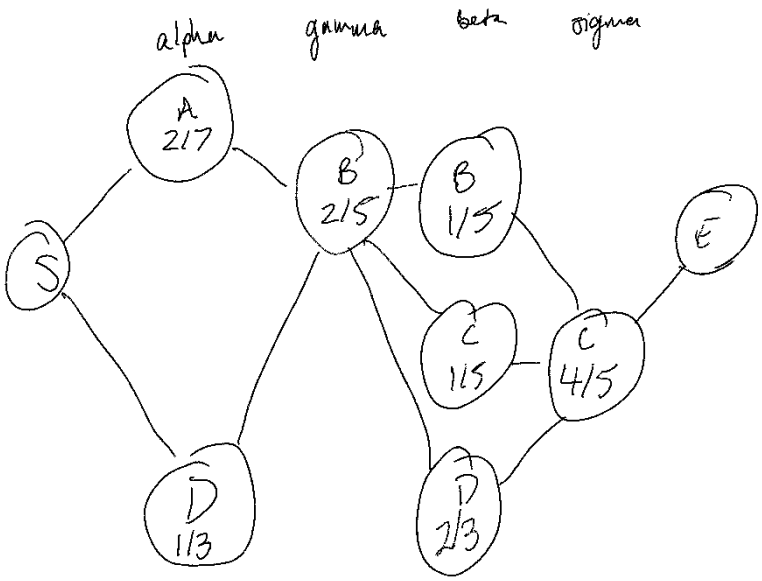
Figure 7: Emission Probabilities

c. How many possible tag sequences (paths) can be generated for the given test sentence? [**Points 3**]
Answer:

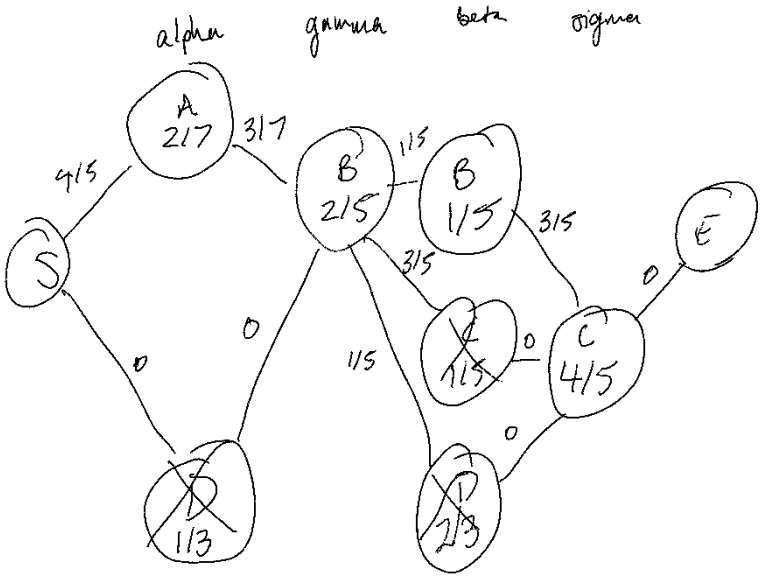
Step 1. Add Emission Probabilities: Using the emission probabilities, we can create the following tag sequence model.



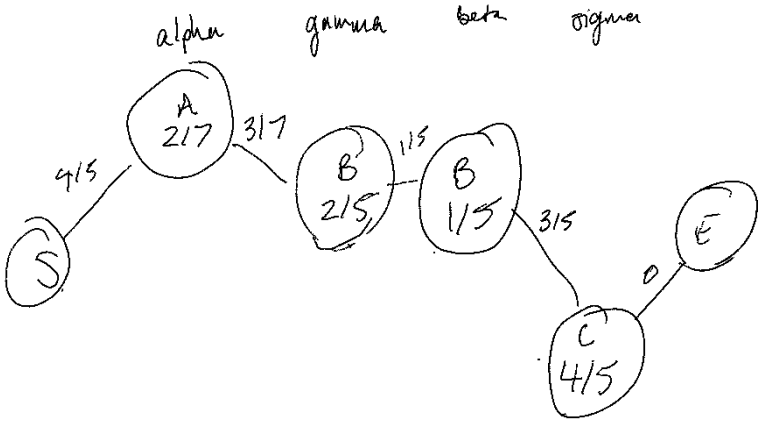
Step 2. Clean-up: Remove all nodes with emission probability of 0.



Step 3. Add Transition Probabilities: Next, we add the transition probabilities.



Step 4. Clean-up: Remove all edges with transition probability of 0 (excluding the only edge leading to the node E). We then remove nodes that do not have a valid sequence that will end in tag E.

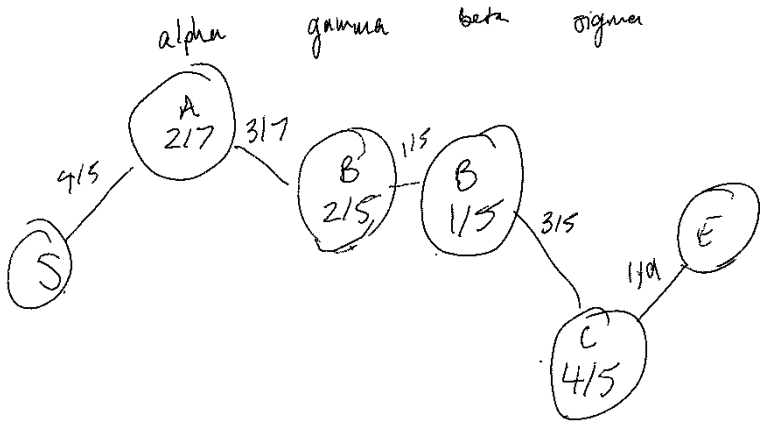


Step 5. Padding:

Since there are no transitions from a tag C (the only possible tag for `sigma` based on the training set) to tag E, **there is only one possible path at this step.** However, the transition probability from the final to the end tag is 0. Since this is unacceptable, we must remedy this in some way. One approach is to pad the transition probability. We can do so by adding 1 to each cell in row C and recalculating the transition probabilities from tag C.

Tag	A	B	C	D	E	totals
S	4/5	1/5	0	0	0	5
A	0	3/7	1/7	0	3/7	7
B	0	1/5	3/5	1/5	0	5
C	**3/9**	**2/9**	**1/9**	**3/9**	**1/9**	**9**
D	1/3	0	0	0	2/3	3

Figure 8: Transition Probabilities **Modified**



d. Assign possible tags for the given test sentence following maximum likelihood calculation. Please show details likelihood calculation for all the optimized tag sequences. [**Points 6**]

Answer:

With the transition probability modified we can now calculate the maximum likelihood probabilities. There is only one path, so there is only one calculation.

$$S \to A \to B \to B \to C \to E =$$
$$\frac{4}{5} * \frac{2}{7} * \frac{3}{7} * \frac{2}{5} * \frac{1}{5} * \frac{1}{5} * \frac{3}{5} * \frac{4}{5} * \frac{1}{9} = 0.00008359$$

Therefore, the assigned tags to the test sentence `alpha gamma beta sigma` are `[A, B, B, C]`. However, this is somewhat unsatisfactory. To allow for more possible tag sequences to be compared, the training set should be expanded.

10. [**Points 10**] A restaurant chain wants to see whether customers like their foods or not. They hire you as an NLP scientist to do this task. You crawled their website, collect and annotate all the reviews. Your task is to conduct the experiment for this text data.

    a. How many classes would you identify for this food review and why? Please list down the class names as your preferences. [**Points 2**]

    Answer:

    Since the chain would like to see whether customers like (*positive*) or dislike (*negative*), we can use the standard approach for sentiment analysis, which utilizes three classes: *positive*, *negative* and *neutral*. However, to allow for a simpler model, we will use binary classification to look for *positive* and *negative* reviews and disregard *neutral* reviews.

    b. You train two classifiers – (i) logistic regression/softmax classifier, and (ii) a two-layer feed forward neural network. Please mention differences between these two classifiers. [**Points 3**]

    Answer:

    A logistic regression/softmax classifier is a linear classifier that uses a logistic function to make predictions. It is a binary classifier, which means that it can only classify two classes, and thus is suitable for this task. The logistic function gives a probability for the class, then a softmax function is used to normalize the output to get the final class prediction.

    A two-layer feed forward neural network is a feed forward neural network that has two hidden layers. The neurons in each hidden layer of the neural network have inputs, outputs, and a differentiable activation function. The neurons are also connected to the next layer through weights and biases. The activation function allows the neuron to take inputs and produce outputs. The output layer is a fully connected layer with a softmax activation function. Since the activation functions are differentiable, we can use backpropagation in order to train the model.

    c. Now you are deciding to pick only one classifier for the deployment environment. How would you design your experimental setup to evaluate these two classifiers? How would you decide to pick which classifier for the deployment environment? Please mention the details reason behind your decision. [**Points 5**]

    Answer:

    By using k-fold cross validation, we can evaluate the average performance of each classifier with the given input data set. It is common to use $k = 10$, but this will depend on the amount of available data. With 10-fold cross validation, first we split the dataset into 10 folds. We set aside 1 fold for testing, and use the remaining 9 folds for training. We then train the classifier on the training folds and test it on the testing fold. We repeat this process 10 times and save the final evaluation score. At the end of training, we can average the evaluation scores and compare the performance of each model. Evaluation scores can include precision, recall, and accuracy.

1 Appendix

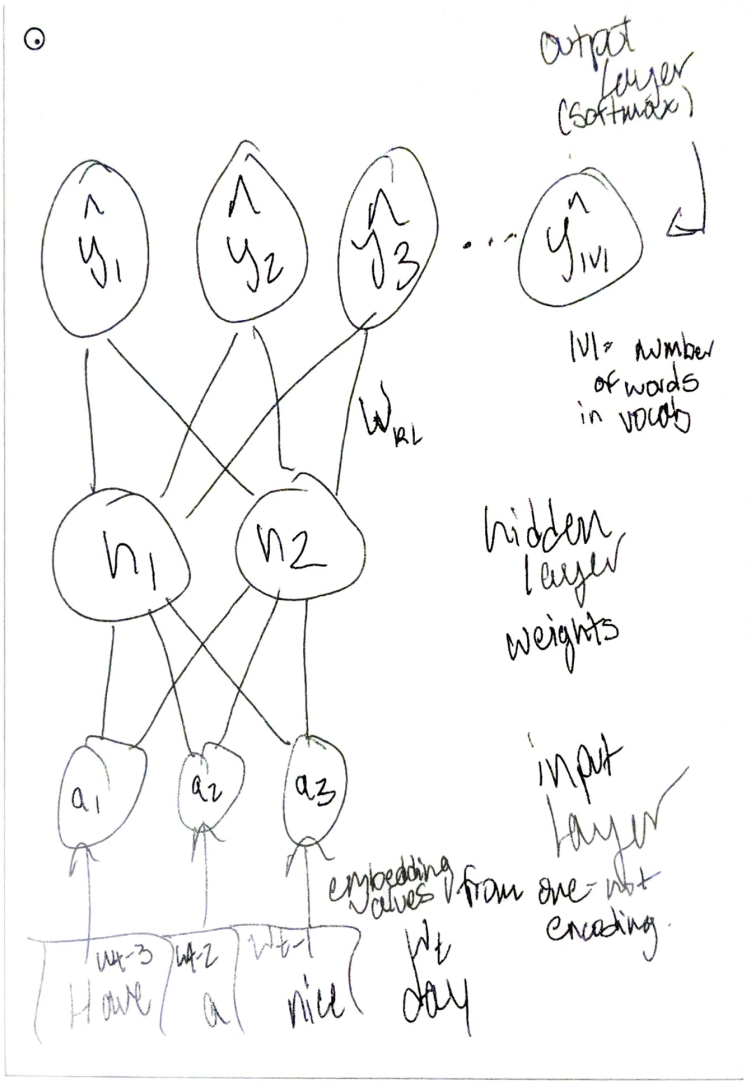


Figure 9: NN-language Model