**Midterm Question-answer**

You are consulting for a healthcare company. They provide you with clinical notes of the first encounter that each patient had with their doctor regarding a particular medical episode. There are a total of 12 million patients and clinical notes. Figure 2 shows a sample clinical note. At the time that each clinical note was written, the underlying illnesses associated with the medical episode were unknown to the doctor. The company provides you with the true set of illnesses associated with each medical episode and asks you to build a model that can infer these underlying illnesses using only the current clinical note and all previous clinical notes belonging to the patient. The set of notes provided to you span 10 years; each patient therefore can have multiple clinical notes (medical episodes) in that period.

A close-up of a list of medical records

Description automatically generated

Each note can contain any number of tokens (see the figure above). Some tokens (e.g. ”Meds”) occur more frequently than others in the collection of notes provided to you.

You call a former student of the NLP class for advice. He tells you to first create a distributed representation of each patient note by combining the distributed representations of the words contained in the note.

Answer ***Questions 10 - 13*** based on these settings:

**Q1) Given the sample note provided in the figure above, how would you map the various tokens into distributed vector representations? Explain the reasoning for your choice of solution approach.**

To map the various tokens in the sample clinical note into distributed vector representations, you can follow these steps:

**1. Tokenization:** First, you would need to tokenize the text. Tokenization is the process of breaking the text into individual words or subword units. You should consider using a pre-trained tokenizer that can handle various languages and subword units, like Byte-Pair Encoding (BPE) or Sentence Piece.

**2. Word Embeddings:** After tokenization, you will obtain a list of tokens. You can use pre-trained word embeddings to convert each token into a fixed-size vector representation. Pre-trained word embeddings, like Word2Vec, GloVe, or FastText, have been trained on a large corpus of text and provide vector representations for common words. You would lookup each token in the embedding vocabulary and replace it with the corresponding vector.

**3. Combine Word Vectors:** Once you have the word vectors for all the tokens in the clinical note, you can combine them to create a distributed representation for the entire note. There are several approaches to do this:

a. Averaging: You can calculate the mean of all the word vectors in the note. This is a simple and effective method, which works well for most cases.

b. Summation: You can sum up all the word vectors. This approach captures the overall context of the note, but it may lead to larger vector representations.

The choice of the solution approach depends on your specific use case and the nature of the clinical notes. For a healthcare application, you may want to experiment with different approaches and evaluate their performance in downstream tasks like disease prediction. It's important to preprocess the text data effectively and choose the approach that best captures the semantic meaning of the clinical notes to improve the accuracy of your disease prediction model.

**Q2) A patient may have any number of illnesses from a list of 70,000 known medical illnesses. The output of your neural network classifier will therefore be a vector with 70,000 elements, where each element in this output vector represents the probability that the patient has the illness that maps to that particular element. Your experienced friend tells you that illnesses are not mutually exclusive - i.e., having one illness does not preclude you from having any other illnesses. Given this insight, is it better to have a sigmoid non-linearity or a softmax non-linearity as your output unit? Why?**

In this scenario, it is better to use a sigmoid non-linearity as the output unit for your neural network classifier, rather than a softmax non-linearity. Here's why:

**Non-Mutually Exclusive Illnesses:** As your experienced friend has mentioned, illnesses are not mutually exclusive. This means that a patient can have multiple illnesses simultaneously. Using a sigmoid non-linearity allows each element in the output vector to represent the probability of a patient having a specific illness independently. Sigmoid activation functions can output values in the range [0, 1] and are well-suited for modeling the probability of binary events, such as whether a patient has a specific illness.

**Softmax vs. Sigmoid:** The softmax activation function is typically used when you have mutually exclusive classes, such as in a multi-class classification problem where each sample belongs to only one class. It normalizes the output vector such that the sum of all class probabilities is equal to 1. In your case, using softmax would imply that a patient can have only one illness, which is not the case.

Using sigmoid activations allows your model to make multiple predictions for each illness independently. This is essential for capturing the possibility of co-occurring illnesses. With sigmoid activation, the sum of the individual illness probabilities for a single patient can be greater than 1, indicating that the patient can have multiple illnesses, which aligns with the real-world scenario you're dealing with. Sigmoid activation functions are more flexible when dealing with multi-label classification problems, where each label (illness) can be present or absent independently. The model can make predictions for each illness without imposing constraints on the presence of other illnesses.

**Q3) You now have a distributed representation of each patient note (note-vector). You assume that a patient's past medical history is informative of their current illness. As such, you are faced with the choice between two solution approaches:**

* **(1) applying a recurrent neural network to predict the current illness based on the patient's current and previous note-vectors; or:**
* **(2) use a window-based feedforward neural network approach, in which your input is the summation (or average) of past and current note-vectors.**

**Which one would you choose for this task? In your answer, compare the pros and cons of each approach, and explain why your choice is better than the alternative.**

**Answer:** In this scenario, I would choose the recurrent neural network (RNN) approach (Option 1) to predict the current illness based on the patient's current and previous note-vectors. Here's a comparison of the two approaches and an explanation of why the RNN approach is better:

**Option 1: Recurrent Neural Network (RNN)**

**Pros:**

1. Sequential Information: RNNs are designed to capture sequential information and dependencies in the data. Since a patient's past medical history is informative of their current illness, RNNs are well-suited for modeling the temporal relationships between different clinical notes.

2. Variable-Length Inputs: RNNs can handle variable-length sequences, which is crucial in the healthcare domain where the number of previous notes may vary from one patient to another.

3. Memory of Past States: RNNs maintain an internal memory of past states, allowing them to consider the entire history of note-vectors when making predictions about the current illness. This is essential in capturing long-term dependencies.

**Cons:**

1. Vanishing Gradient: RNNs can suffer from vanishing gradient problems, which may lead to difficulties in capturing long-range dependencies in sequences. However, this can be partially mitigated with variants like Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU).

**Option 2: Window-Based Feedforward Neural Network**

**Pros:**

1. Simplicity: A feedforward neural network is conceptually simpler to implement and train compared to RNNs. It doesn't require modeling sequential dependencies explicitly.

2. Parallelism: Feedforward networks can take advantage of parallel processing, potentially leading to faster training and inference.

**Cons:**

1. Loss of Sequential Information: The window-based approach loses the temporal order of clinical notes. It treats all notes within a window as equally important, which may not be suitable for capturing the dynamic changes in a patient's health over time.

2. Fixed Window Size: Determining the appropriate window size can be challenging, as it doesn't account for varying numbers of previous notes among patients. Choosing a small window may result in information loss, while a large window may introduce noise.

3. Difficulty in Capturing Long-Term Dependencies: Feedforward networks struggle with capturing long-term dependencies in sequential data, which is essential when considering a patient's entire medical history.

**Why I Choose Option 1 (RNN):**

Given that a patient's past medical history is informative and that clinical notes are sequential in nature, the RNN approach is better for several reasons:

1. Sequential Information: RNNs are specifically designed to handle sequential data and are well-suited to capture the temporal dependencies in clinical notes.

2. Variable-Length Sequences: Healthcare data often involves varying numbers of clinical notes per patient. RNNs can handle this variability, ensuring that all available information is considered.

3. Memory of Past States: RNNs maintain memory of past states, allowing them to model the entire patient history effectively.

In summary, RNNs are more suitable for modeling the sequential nature of clinical notes and capturing the temporal relationships between notes, making them the preferred choice for this task compared to the window-based feedforward approach.

**Q4) You try to figure out a better way to reduce the training and testing time of your model. You perform a run time analysis and observe that the computational bottleneck is in your output unit: the number of target illnesses is too high. Your expert friend tells you that each illness in the list of 70,000 illnesses belongs to one of 300 classes (e.g. a migraine belongs to the neurological disorder class). He shares with you a dictionary which maps each illness to its corresponding class. How can you use this information to reduce the time complexity of your model? Include your forward propagation equations in your answer.**

To reduce the training and testing time of your model, you can use the fact that each of the 70,000 illnesses belongs to one of 300 classes. This is like grouping similar illnesses together.

Here's a simple way to do it:

**First Stage: Class Prediction**

First, your model predicts which of the 300 classes an illness belongs to.

For example, it decides if the illness is related to "heart problems" or "skin conditions."

**Second Stage: Illness Prediction within Class**

Once the class is known, your model predicts the specific illness within that class.

If the class is "heart problems," it figures out if it's "hypertension" or "arrhythmia."

**Final Prediction**

You combine the class and specific illness predictions to get the final result.

For example, the model might say, "This illness belongs to the 'heart problems' class, and it's most likely 'hypertension.'"

This two-stage approach reduces the time your model needs to make predictions because it narrows down the possibilities in the first step, making it faster and more efficient.

Here are the forward propagation equations for each stage of the hierarchical classification:

**First Stage: Class Prediction**

- Input Note-Vector: X

- Weight Matrix and Bias for Class Prediction: W\_class, b\_class

- Calculate the class scores (Z\_class) for each of the 300 classes:

Z\_class = X \* W\_class + b\_class

- Apply softmax activation to obtain class probabilities (P\_class):

P\_class = softmax(Z\_class)

**Second Stage: Illness Prediction within Class**

- Given the predicted class C (e.g., "heart problems"), use a specific set of weights and biases for that class.

- Input Note-Vector: X

- Weight Matrix and Bias for Illness Prediction within Class C: W\_illness\_C, b\_illness\_C

- Calculate the illness scores (Z\_illness) within the selected class C:

Z\_illness\_C = X \* W\_illness\_C + b\_illness\_C

- Apply sigmoid activation for each illness to obtain individual illness probabilities within the class (P\_illness\_C).

**Final Prediction**

- To make the final prediction, select the class with the highest probability from P\_class as the predicted class.

- For the selected class C, choose the illness with the highest probability from P\_illness\_C for that class.

These forward propagation equations represent the flow of information in your two-stage hierarchical classification model. The first stage determines the class, and the second stage predicts the specific illness within that class, ultimately providing a more efficient and accurate way to handle a large number of potential illnesses.