2 Maximum Likelihood Estimation

Our code for Part 2 is within the files *Part2.py* and *Part2.ipynb*.

2.1 Emission Parameters Estimation

In order to derive the count values, we process the file as follows.
Each line in the file represents a word followed by its label. For each label, we store the number of occurrences of each word in a dictionary with the key as the word, and the value as the number of occurrences. For a particular label, this results in a dictionary that looks something like the following:

```
{'Nous': 100, 'avons': 101, 'tout': 105, 'aim': 7, '.': 1067, ... }
```

We store this dictionary as the value of another dictionary, where the key represents the label. This results in the following dictionary:

```
emissions = {'B-negative': {'"': 1, 'Accueil': 7, 'Ambiance': 2, ...},
             'B-positive': { ... }, ... }
```

This allows us to efficient retrieve Count($y \rightarrow x$) using

```
emissions[y][x]
```

Similarly, for computational efficiency, we store the counts of each label in a separate dictionary:

```
emission_counts = {'B-negative': 675, 'B-neutral': 113, 'B-positive': 810, 'I-negative': 2
```

We can then easily retrieve Count(y) using

```
emission_counts[x]
```

This allows us to efficiently retrieve the emission parameters $e(x|y)$, at the cost of a minor space complexity increase.

2.2 Laplace Smoothing

For smoothing, we implement a getter function for the emission parameters. If the given word x does not exist in our parameters learnt from training data, we return the smoothed version of the parameters.

```
def get_emission_parameters(emissions, emission_counts, x, y, k=1):
    ''' Returns the MLE of the emission parameters based on the emissions dictionary '''
    state_data = emissions[y]
    count_y = emission_counts[y] #sum(state_data.values()) # Denominator

    # If x == "#UNK#", it will return the following
    count_y_x = k

    # If x exists in training, return its MLE instead
    if x != "#UNK#":
        count_y_x = state_data[x] # Numerator

    e = count_y_x / (count_y + k)
    return e{}
```

2.3 Sequence Labeling

For sequence labeling, we implement a function that takes as input the sentence to be labeled, and the emission parameters as described in Section 2.1:

```
def label_sequence(sentence, emissions, emission_counts):
    ''' Takes a list `sentence` that contains words of a sentence as strings '''
    tags = []

    for word in sentence:
        predicted_label = ""
        max_prob = -1

        # Find y with maximum probability
        for y in emissions:

            if word not in observations:
                word = "#UNK#"

            if (word in emissions[y]) or (word == "#UNK#"):
                prob = get_emission_parameters(emissions, emission_counts, word, y)

                # If this is higher than the previous highest, use this
                if prob > max_prob:
                    predicted_label = y
                    max_prob = prob

        # Add prediction to list
        tags.append(predicted_label)

    return tags
```

For each word, it finds the arg max over y, and assigns this as the tag. The final output of this function is a list containing the predicted tags.

2.4 Results

Our results on the four datasets are as follows:

Dataset	**F-Score**	
	Entity	**Entity Type**
EN	0.6297	0.4595
SG	0.2952	0.1501
CN	0.1929	0.1134
FR	0.2751	0.1169

3 First-order hidden Markov model

4 Second-order hidden Markov model

5 Design challenge

5.1 Preprocessing

5.1.1 Tokenizing

Each file can be considered a corpus, the documents being each sentence in the file. Tokenizing was carried out by converting each sentence into a list of words. During tokenization, all whitespace was stripped from the beginning and end of each word. Also, each token is normalized by converting to lowercase.

5.1.2 Replacing irrelevant words

In the initial problem, the training data contains a large vocabulary of words which occurred only a single time. To mitigate this issue, several types of words were converted into various placeholders instead, as they do not carry semantic meaning.

The placeholders used are as follows:

1. `#PUNC#`: strings that do not contain alphanumeric characters
2. `#HASH#`: hashtags (strings beginning with a `#` character)
3. `#AT#`: `@` mentions (strings beginning with a `@` character)
4. `#NUM#`: strings that contain only digits
5. `#URL#`: strings that begin with `http:` and end with `.com`

5.1.3 Stop words

Stop words were also initially converted to a token labeled `#STOP#`. However, during evaluation on the `EN` and `SG` dataset with Part 3, it was shown to reduce the F_1 score on both the *Correct Entity* and *Correct Entity Type* tasks. It was also shown to produce poorer F_1 scores on the `EN` dataset with Part 5.

Taking into account the negative correlation between removing stop words and F_1 scores, as well as the fact that we do not have French stop words, we decided not to remove stop words from the training data.

5.2 Inputs and labels

5.3 RNN

5.3.1 Forward

We use a RNN, with a computational graph as shown in Figure 1. The RNN maps an input sequence $\boldsymbol{x}$ to an output sequence $\boldsymbol{o}$. The predicted labels are then given by $\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{o})$.
The loss function L compares the predicted labels $\hat{\boldsymbol{y}}$ with the target labels $\boldsymbol{y}$ through cross-entropy loss, where C is the number of classes in the classification task.

$$L = -\sum_{i}^{C} \boldsymbol{y}_i \cdot \log(\hat{\boldsymbol{y}}_i)$$

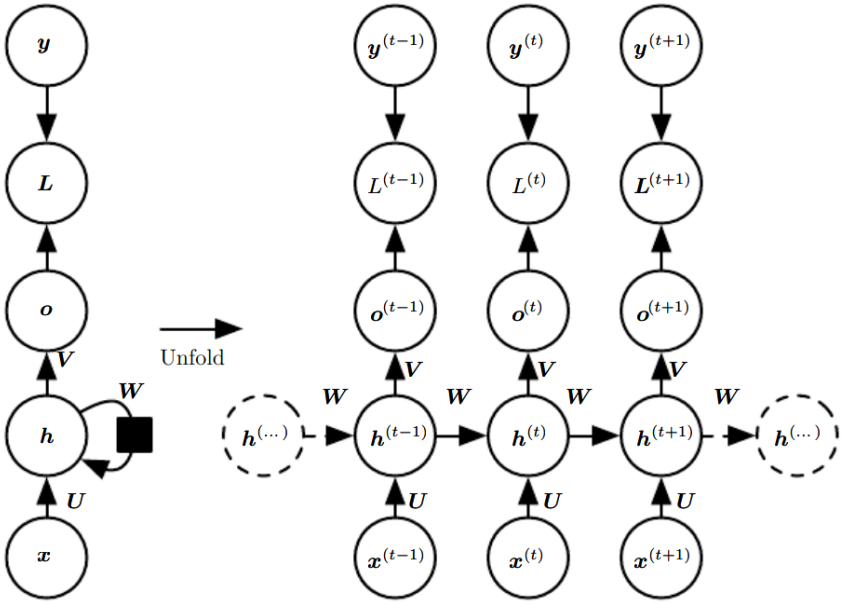

Figure 1: RNN

The remaining nodes in the graph are computed as shown in Equation 1.

$$\begin{aligned} \boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{U}\boldsymbol{x}^{(t)} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} \\ \boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}) \\ \boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)} \\ \hat{\boldsymbol{y}}^{(t)} &= \text{softmax}(\boldsymbol{o}^{(t)}) \end{aligned} \tag{1}$$

$\boldsymbol{U}, \boldsymbol{W}, \boldsymbol{V}$ are the weight matrices, while $\boldsymbol{b}, \boldsymbol{c}$ are biases. By choice of hyperparameters, $\boldsymbol{h}^{(t)} \in \mathbb{R}^{128\times 1}$.

For an input sequence, $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, ..., \boldsymbol{x}^{(n)}$, we first compute the hidden units $\boldsymbol{h}^{(t)}$ as shown in Equation 2. At $t = 1$, the hidden unit is computed using only $x^{(1)}$. Once we have the value of $\boldsymbol{h}^{(t)}$, we can then compute its predicted state $\hat{\boldsymbol{y}}^{(t)}$ with Equation 1.

$$\begin{aligned} \boldsymbol{h}^{(1)} &= \tanh(\boldsymbol{b} + \boldsymbol{U}\boldsymbol{x}^{(1)}) \\ \boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{b} + \boldsymbol{U}\boldsymbol{x}^{(t)} + \boldsymbol{W}\boldsymbol{h}^{(t-1)}),\ \forall\, t \in \{2, ..., n\} \end{aligned} \tag{2}$$

5.3.2 Backward

To perform backpropagation, our objective is to obtain the partial differential of L with respect to the weights and biases. The gradients are taken to be the **sum** over all time steps $t \in \{1, ..., n\}$.

Output layer

$$\begin{aligned} \frac{\partial L}{\partial \boldsymbol{o}^{(t)}} &= \text{softmax}(\boldsymbol{o}^{(t)}) - \boldsymbol{y}^{(t)} \\ &= \hat{\boldsymbol{y}}^{(t)} - \boldsymbol{y}^{(t)} \end{aligned} \tag{3}$$

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{c}} &= \sum_{t=1}^{n} \frac{\partial L}{\partial \boldsymbol{o}^{(t)}} \frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{c}} \\ &= \sum_{t=1}^{n} (\hat{\boldsymbol{y}}^{(t)} - \boldsymbol{y}^{(t)}) \frac{\partial (\boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)})}{\partial \boldsymbol{c}} \\ &= \sum_{t=1}^{n} (\hat{\boldsymbol{y}}^{(t)} - \boldsymbol{y}^{(t)})\end{aligned} \tag{4}$$

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{V}} &= \sum_{t=1}^{n} \frac{\partial L}{\partial \boldsymbol{o}^{(t)}} \frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{V}} \\ &= \sum_{t=1}^{n} (\hat{\boldsymbol{y}}^{(t)} - \boldsymbol{y}^{(t)}) \frac{\partial (\boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)})}{\partial \boldsymbol{V}} \\ &= \sum_{t=1}^{n} (\hat{\boldsymbol{y}}^{(t)} - \boldsymbol{y}^{(t)})\ {\boldsymbol{h}^{(t)}}^{T}\end{aligned} \tag{5}$$

Hidden layer

To compute the gradients of the hidden units $\frac{\partial L}{\partial \boldsymbol{h}^{(t)}}$, we initialize the gradient at $t = n$, since it has no subsequent time step, and its gradient is dependent only on $\boldsymbol{o}^{(n)}$.

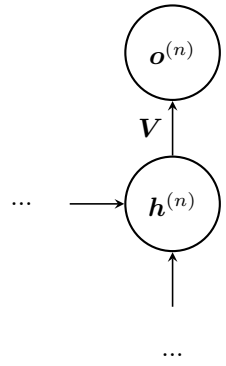

Figure 2: Computational graph showing gradients for $\boldsymbol{h}^{(n)}$

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{h}^{(n)}} &= \frac{\partial L}{\partial \boldsymbol{o}^{(n)}} \frac{\partial \boldsymbol{o}^{(n)}}{\partial \boldsymbol{h}^{(n)}} \\ &= (\hat{\boldsymbol{y}}^{(n)} - \boldsymbol{y}^{(n)}) \frac{\partial (\boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(n)})}{\partial \boldsymbol{h}^{(n)}} \\ &= \boldsymbol{V}^{T}(\hat{\boldsymbol{y}}^{(n)} - \boldsymbol{y}^{(n)})\end{aligned} \tag{6}$$

Now, we can formulate the rest of the gradients recursively.

$$\frac{\partial L}{\partial \boldsymbol{h}^{(t)}} = \frac{\partial L}{\partial \boldsymbol{o}^{(t)}} \frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{h}^{(t)}} + \frac{\partial L}{\partial \boldsymbol{h}^{(t+1)}} \frac{\partial \boldsymbol{h}^{(t+1)}}{\partial \boldsymbol{h}^{(t)}}$$

From the result in Equation 6, we have that

$$\frac{\partial L}{\partial \boldsymbol{o}^{(t)}} \frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{h}^{(t)}} = \boldsymbol{V}^{T}(\hat{\boldsymbol{y}}^{(t)} - \boldsymbol{y}^{(t)})$$

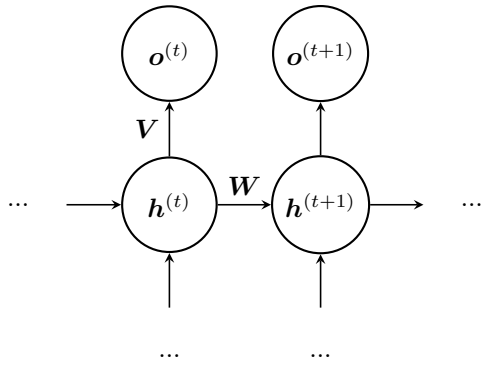

Figure 3: Computational graph showing gradients for $\boldsymbol{h}^{(t)}$

$$\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{h}^{(t+1)}}\frac{\partial \boldsymbol{h}^{(t+1)}}{\partial \boldsymbol{h}^{(t)}} &= \frac{\partial L}{\partial \boldsymbol{h}^{(t+1)}}\frac{\partial \boldsymbol{h}^{(t+1)}}{\partial \boldsymbol{a}^{(t+1)}}\frac{\partial \boldsymbol{a}^{(t+1)}}{\partial \boldsymbol{h}^{(t)}} \\
&= \frac{\partial L}{\partial \boldsymbol{h}^{(t+1)}}\frac{\partial(\tanh(\boldsymbol{a}^{(t+1)}))}{\partial \boldsymbol{a}^{(t+1)}}\frac{\partial(\boldsymbol{b}+\boldsymbol{U}\boldsymbol{x}^{(t+1)}+\boldsymbol{W}\boldsymbol{h}^{(t)})}{\partial \boldsymbol{h}^{(t)}} \\
&= \boldsymbol{W}^T\frac{\partial L}{\partial \boldsymbol{h}^{(t+1)}}\frac{\partial(\tanh(\boldsymbol{a}^{(t+1)}))}{\partial \boldsymbol{a}^{(t+1)}} \\
&= \boldsymbol{W}^T\mathrm{diag}(1-(\boldsymbol{h}^{(t+1)})^2)\frac{\partial L}{\partial \boldsymbol{h}^{(t+1)}}
\end{aligned}$$

Combining the above results, we obtain

$$\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{h}^{(t)}} &= \frac{\partial L}{\partial \boldsymbol{o}^{(t)}}\frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{h}^{(t)}} + \frac{\partial L}{\partial \boldsymbol{h}^{(t+1)}}\frac{\partial \boldsymbol{h}^{(t+1)}}{\partial \boldsymbol{h}^{(t)}} \\
&= \boldsymbol{V}^T(\hat{\boldsymbol{y}}^{(t)}-\boldsymbol{y}^{(t)}) + \boldsymbol{W}^T\mathrm{diag}(1-(\boldsymbol{h}^{(t+1)})^2)\frac{\partial L}{\partial \boldsymbol{h}^{(t+1)}}
\end{aligned} \tag{7}$$

It is next imperative to show that the Jacobian $J^{(t+1)} = \frac{\partial \boldsymbol{h}^{(t+1)}}{\partial \boldsymbol{a}^{(t+1)}} = \frac{\partial(\tanh(\boldsymbol{a}^{(t+1)}))}{\partial \boldsymbol{a}^{(t+1)}} = \mathrm{diag}(1-(\boldsymbol{h}^{(t+1)})^2)$.

We begin with the knowledge that $\frac{d}{dx}\tanh(x) = 1-\tanh^2(x)$.

$$J_{ij}^{(t+1)} = \frac{\partial \boldsymbol{h}_i^{(t+1)}}{\partial \boldsymbol{a}_j^{(t+1)}}$$

For $i \neq j$, $J_{ij}^{(t)} = 0$.
For $i = j$,

$$\begin{aligned}
J_{ii}^{(t+1)} = \frac{\partial \boldsymbol{h}_i^{(t+1)}}{\partial \boldsymbol{a}_i^{(t+1)}} &= \frac{\partial(\tanh(\boldsymbol{a}^{(t+1)})_i)}{\partial \boldsymbol{a}_i^{(t+1)}} \\
&= 1-\tanh^2(\boldsymbol{a}^{(t+1)})_i \\
&= 1-(\boldsymbol{h}_i^{(t+1)})^2
\end{aligned}$$

$$\therefore J^{(t)} = \frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{a}^{(t)}} = \begin{pmatrix} 1-(\boldsymbol{h}_1^{(t)})^2 & 0 & ... & 0 \\ 0 & 1-(\boldsymbol{h}_2^{(t)})^2 & ... & 0 \\ ... & ... & 1-(\boldsymbol{h}_i^{(t)})^2 & ... \\ 0 & 0 & ... & 1-(\boldsymbol{h}_n^{(t)})^2 \end{pmatrix} = \mathrm{diag}(1-(\boldsymbol{h}^{(t)})^2)$$

We can then use this result to obtain the gradients with respect to $\boldsymbol{b}, \boldsymbol{W}, \boldsymbol{U}$.

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{b}} &= \sum_{t=1}^{n} \frac{\partial L}{\partial \boldsymbol{h}^{(t)}} \frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{a}^{(t)}} \frac{\partial \boldsymbol{a}^{(t)}}{\partial \boldsymbol{b}} \\ &= \sum_{t=1}^{n} \operatorname{diag}(1 - (\boldsymbol{h}^{(t)})^2) \frac{\partial L}{\partial \boldsymbol{h}^{(t)}} \frac{\partial (\boldsymbol{b} + \boldsymbol{U}\boldsymbol{x}^{(t)} + \boldsymbol{W}\boldsymbol{h}^{(t-1)})}{\partial \boldsymbol{b}} \\ &= \sum_{t=1}^{n} \operatorname{diag}(1 - (\boldsymbol{h}^{(t)})^2) \frac{\partial L}{\partial \boldsymbol{h}^{(t)}}\end{aligned} \tag{8}$$

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{W}} &= \sum_{t=1}^{n} \frac{\partial L}{\partial \boldsymbol{h}^{(t)}} \frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{a}^{(t)}} \frac{\partial \boldsymbol{a}^{(t)}}{\partial \boldsymbol{W}} \\ &= \sum_{t=1}^{n} \operatorname{diag}(1 - (\boldsymbol{h}^{(t)})^2) \frac{\partial L}{\partial \boldsymbol{h}^{(t)}} \frac{\partial (\boldsymbol{b} + \boldsymbol{U}\boldsymbol{x}^{(t)} + \boldsymbol{W}\boldsymbol{h}^{(t-1)})}{\partial \boldsymbol{W}} \\ &= \sum_{t=1}^{n} \operatorname{diag}(1 - (\boldsymbol{h}^{(t)})^2) \frac{\partial L}{\partial \boldsymbol{h}^{(t)}} \boldsymbol{h}^{(t-1)^T}\end{aligned} \tag{9}$$

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{U}} &= \sum_{t=1}^{n} \frac{\partial L}{\partial \boldsymbol{h}^{(t)}} \frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{a}^{(t)}} \frac{\partial \boldsymbol{a}^{(t)}}{\partial \boldsymbol{U}} \\ &= \sum_{t=1}^{n} \operatorname{diag}(1 - (\boldsymbol{h}^{(t)})^2) \frac{\partial L}{\partial \boldsymbol{h}^{(t)}} \frac{\partial (\boldsymbol{b} + \boldsymbol{U}\boldsymbol{x}^{(t)} + \boldsymbol{W}\boldsymbol{h}^{(t-1)})}{\partial \boldsymbol{U}} \\ &= \sum_{t=1}^{n} \operatorname{diag}(1 - (\boldsymbol{h}^{(t)})^2) \frac{\partial L}{\partial \boldsymbol{h}^{(t)}} \boldsymbol{x}^{(t)^T}\end{aligned} \tag{10}$$

In summary,

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{U}} &= \sum_{t=1}^{n} \operatorname{diag}(1 - (\boldsymbol{h}^{(t)})^2) \frac{\partial L}{\partial \boldsymbol{h}^{(t)}} \boldsymbol{x}^{(t)^T} \\ \frac{\partial L}{\partial \boldsymbol{W}} &= \sum_{t=1}^{n} \operatorname{diag}(1 - (\boldsymbol{h}^{(t)})^2) \frac{\partial L}{\partial \boldsymbol{h}^{(t)}} \boldsymbol{h}^{(t-1)^T} \\ \frac{\partial L}{\partial \boldsymbol{b}} &= \sum_{t=1}^{n} \operatorname{diag}(1 - (\boldsymbol{h}^{(t)})^2) \frac{\partial L}{\partial \boldsymbol{h}^{(t)}} \\ \frac{\partial L}{\partial \boldsymbol{V}} &= \sum_{t=1}^{n} (\hat{\boldsymbol{y}}^{(t)} - \boldsymbol{y}^{(t)})\ \boldsymbol{h}^{(t)^T} \\ \frac{\partial L}{\partial \boldsymbol{c}} &= \sum_{t=1}^{n} (\hat{\boldsymbol{y}}^{(t)} - \boldsymbol{y}^{(t)})\end{aligned} \tag{11}$$

5.3.3 Training

5.3.4 Evaluation

5.3.5 Finetuning

5.3.6 Summary