Declaration

1A. Declaration of Original Work.

By entering my student ID below, I certify that I completed my assignment independently of all others (except somewhere sanctioned during in-class sessions), obeying the class policy outlined in the introductory lecture. In particular, I am allowed to discuss the problems and solutions in this assignment, but have waited at least 30 minutes by doing other activities unrelated to class before attempting to complete or modify my answers as per the Pokemon Go rule.

Signed,
A0184679H

2. References

- Choosing Evaluation Metrics For Classification Model (analyticsvidhya.com), for understanding the relationship between binary confusion matrix and evaluation metrics
- differentiate log(e^(2x)) - Wolfram|Alpha, to confirm my calculations of derivatives.

1.1

S = "I don't dislike it"

P(+ | S) = P (+) * P (I | +) * P (don't | +) * P (dislike | +)

P(- | S) = P (-) * P (I | -) * P (don't | -) * P (dislike | -)

P(+) = $\frac{2}{5}$

P(-) = $\frac{3}{5}$

P(I | +) = $\frac{3}{18}$

P (don't | +) = $\frac{2}{18}$

P (dislike | +) = $\frac{2}{18}$

P (I | -) = $\frac{4}{22}$

P (don't | -) = $\frac{2}{22}$

P (dislike | -) = $\frac{3}{22}$

P(+ | S) = $\frac{2}{5} * \frac{3}{18} * \frac{2}{18} * \frac{2}{18}$ = 0.000823

P(- | S) = $\frac{3}{5} * \frac{4}{22} * \frac{2}{22} * \frac{3}{22}$ = 0.001352

∵ P(- | S) > P(+ | S), the label for "I don't dislike it" is negative.

1.2

There are various ways to improve the accuracy of the Naive Bayes' classifier, including preprocessing such as stemming, lemmatization, and stopwords removal such as "I", feature engineering such as using bag of words, tf-idf. I will use one of the feature engineering techniques, the n-gram (bigrams in particular) with add 1 smoothing to improve the model's accuracy.

bigrams	+	-
I don't	1	1
don't like	0	1
like the	1	1
the food	1	1
I dislike	0	1
dislike the	1	1
the burger	0	1
the manager	0	1
I like	1	0
the food	1	0
don't dislike	1	0
the ambience	1	0
<s>I	2	3
food</s>	1	1
burger</s>	0	1
manager</s>	0	1
ambience</s>	1	0

P(+|S) = P(+)*P(<s>I|+) * P(I don't|+) * P(don't dislike|+)

$= \frac{2}{5} * \frac{3}{12+18} * \frac{2}{12+18} * \frac{2}{12+18}$

= 0.0001778

P(+|S) = P(+)*P(<s>I|+) * P(I don't|+) * P(don't dislike|+)

$= \frac{3}{5} * \frac{4}{14+18} * \frac{2}{12+18} * \frac{1}{12+18}$

= 0.000146

∵ P(+ | S) > P(- | S), the label for "I don't dislike it" is now correctly labelled as positive.

1.3

When performing binary classification tasks, four results are possible:

- True positive. The model correctly predicts the sample, and the sample is indeed positive.
- True negative. The model correctly predicts the sample, and the sample is indeed negative.
- False positive. The model incorrectly predicts that the sample to be positive, but the sample is indeed negative.
- False negative. The model incorrectly predicts the sample to be negative, but the sample is indeed positive.

Referring to the binary confusion matrix, each outcome is placed as follows:

	Actual positive	Actual negative
Predicted positive	True positive	False positive
Predicted negative	False negative	True negative

Additionally, 3 metrics are defined to evaluate the model: recall, precision and accuracy. Their definition as follows:

Recall: $\frac{true\ positive}{true\ positive + false\ negative}$

Precision: $\frac{true\ positive}{true\ positive + false\ positive}$

Accuracy: $\frac{true\ positive + true\ negative}{true\ positive + false\ negative + false\ positive + false\ negative}$

To reduce the number of positive samples identified as negative, we want to find out the coverage of predicted positive values versus the actual predicted values. As predicted values grow closer to the actual values, the model is performing well according to the evaluating metric. According to the binary confusion matrix, the actual positive value is equal to true positive + predicted positive, therefore recall is the suitable evaluating metric. On the other hand, minimizing the number of predicted positives, say, predicting all samples to be negative, would result in a recall score of 0, and labelling each sample to be negative will result in a recall of 1.

2.1

The input layer has D_x input neurons and a bias, the hidden layer has H input neurons and a bias, and the output vector has D_y neurons. Therefore the total number of parameters is

$$(D_x + 1) * H + (H + 1) * D_y$$

2.2

$$\sigma(x) = \frac{1}{1+e^{-x}} = (1 + e^{-x})^{-1}$$

$$\begin{aligned}
\frac{\partial}{\partial x}\sigma(x) &= -(1 + e^{-x})^{-2} * \frac{\partial}{\partial x}(1 + e^{-x}) \\
&= -(1 + e^{-x})^{-2} * e^{-x} * \frac{\partial}{\partial x}(-x) \\
&= \frac{e^{-x}}{(1+e^{-x})^2} \\
&= \frac{(1+e^{-x}) - 1}{(1+e^{-x})^2} \\
&= \frac{1+e^{-x}}{(1+e^{-x})^2} - \frac{1}{(1+e^{-x})^2} \\
&= \frac{1}{(1+e^{-x})} - \frac{1}{(1+e^{-x})^2} \\
&= \sigma(x) - \sigma(x)^2 \\
&= \sigma(x)\,(1 - \sigma(x))
\end{aligned}$$

2.3

cross entropy: $L_{CE}(y, \hat{y}) = \Sigma_i y_i (log\, \hat{y}_i)$

softmax(θ): $\frac{e^{\theta}}{\Sigma_{j=1}^{k} e^{\theta j}}$

Two cases: i = j and i != j.

Case 1: i = j

$$\frac{\partial \hat{y}_i}{\partial \theta_i} = \frac{\partial}{\partial \theta_i}\left(\frac{e^{\theta}}{\Sigma_{j=1}^{k} e^{\theta j}}\right)$$

$$= \frac{e^{\theta_i}(\Sigma_{j=1}^{k} e^{\theta_j}) - e^{\theta_i}(e^{\theta_i})}{(\Sigma_{j=1}^{k} e^{\theta_j})^2} \quad \left(\text{differentiation, } \frac{f'(x)g(x) - f(x)g'(x)}{(g(x))^2}\right)$$

$$= \frac{e^{\theta_i}(\Sigma_{j=1}^{k} e^{\theta_j} - e^{\theta_i})}{(\Sigma_{j=1}^{k} e^{\theta_j})^2}$$

$$= \frac{e^{\theta_i}}{\Sigma_{j=1}^{k} e^{\theta_j}} * \frac{\Sigma_{j=1}^{k} e^{\theta_j}}{(\Sigma_{j=1}^{k} e^{\theta_j}} - \frac{e^{\theta_i}}{\Sigma_{j=1}^{k} e^{\theta_j}}$$

$$= \hat{y}_i (1 - \hat{y}_i) \quad (\hat{y}_i \text{ is the softmax function of } \theta)$$

Case 2: i != j

$$\frac{\partial \hat{y}_i}{\partial \theta_i} = \frac{\partial}{\partial \theta_i}\left(\frac{e^{\theta}}{\Sigma_{j=1}^{k} e^{\theta j}}\right)$$

$$= \frac{0 * \Sigma_{j=1}^{k} e^{\theta j} - e^{\theta_i}(e^{\theta_j})}{(\Sigma_{j=1}^{k} e^{\theta_j})^2}$$

$$= \frac{-e^{\theta_i}(e^{\theta_j})}{(\Sigma_{j=1}^{k} e^{\theta_j})^2}$$

$$= -\hat{y}_i \hat{y}_j$$

$$\frac{\partial L_{CE}(y, \hat{y})}{\partial \hat{y}_i} = \frac{\partial}{\partial \theta_i}(-\Sigma_i y_i (log\, \hat{y}_i))$$

$$= -\Sigma_i y_i \left(\frac{1}{\hat{y}_i}\right)$$

$$\frac{\partial L_{CE}(y, \hat{y})}{\partial \theta_i} = -\Sigma_{i \neq j} y_j \left(\frac{1}{\hat{y}_j} * \frac{\partial \hat{y}_i}{\partial \theta_i}\right) - y_i \left(\frac{1}{\hat{y}_j} * \frac{\partial \hat{y}_j}{\partial \theta_i}\right)$$

$$= -\Sigma_{i \neq j} y_j \left(\frac{1}{\hat{y}_j} * (-\hat{y}_i \hat{y}_j)\right) - y_i \left(\frac{1}{\hat{y}_j} * \hat{y}_i (1 - \hat{y}_i)\right)$$

$$= \Sigma_{i \neq j} y_j \hat{y}_i + y_j \hat{y}_i - y_i$$

$$= \Sigma_j y_j \hat{y}_i - y_i$$

$$= \hat{y}_i - y_i \quad \text{(one hot label vector, sum of j terms is 1)}$$

3.1

P (into | crises), P (banking | crises), P (as | crises), P (the | crises)

3.2

P (problems | turning), P (into | turning)

3.3

The minimum number of vector dot product operations required is 100. The numerator of $P(O = o \mid C = c) = \frac{e^{(v_o * v_c)}}{\sum_{w \in vocab} e^{(v_w * v_c)}}$ can be calculated once and cached / memoized in a variable. For 100 different words in the vocabulary, and to calculate $P(v_{turning} \mid v_{banking})$, $P(v_{into} \mid v_{banking})$, $P(v_{as} \mid v_{banking})$, $P(v_{crises} \mid v_{banking})$, since only O is different for each conditional probability, the cached value can be used repeatedly for each probability calculation.

3.4

$$P(O = o \mid C = c) = \frac{e^{(v_o * v_c)}}{\Sigma_{w \in vocab} e^{(v_w * v_c)}}$$

$$L_{naive-softmax}(v_o, v_c) = -log\left(\frac{e^{(v_o * v_c)}}{\Sigma_{w \in vocab} e^{(v_w * v_c)}}\right)$$

$$= -(v_o * v_c) - log\left(\Sigma_w e^{(v_w * v_c)}\right)$$

$$\frac{\partial L_{naive-softmax}(v_o, v_c)}{\partial v_c} = \frac{\partial}{\partial v_c}\left(-(v_o * v_c) - log\left(\Sigma_w e^{(v_w * v_c)}\right)\right)$$

$$= -v_o + \frac{\partial}{\partial v_c}\left(log\left(\Sigma_{w \in vocab} e^{(v_w * v_c)}\right)\right)$$

$$= -v_o + \Sigma_{w' \in vocab} \frac{e^{(v_{w'} * v_c)}}{\Sigma_{w \in vocab} e^{(v_w * v_c)}} * v_w$$

$$= -v_o + v_w * \Sigma_{w' \in vocab} \frac{e^{(v_{w'} * v_c)}}{\Sigma_{w \in vocab} e^{(v_w * v_c)}}$$

From the formula, it is inefficient as we will need to calculate the dot product for each word in the vocabulary with the context word c. Unlike question 3.3, this value cannot be memoized and reused, as the result is dependent on v_c, and v_c is different for the loss function for each pair of (context word, center word). Furthermore, the double summation symbol used in the loss function derivative indicates that some form of double looping throughout the vocabulary will need to be implemented, therefore it is inefficient.

3.5

The number of dot product operations required when K = 5 is 6. According to the function

$$L_{neg-sample}(v_o, v_c) = -log(\sigma(v_o * v_c)) - \Sigma_{k=1}^{K} log(\sigma(-v_k * v_c))$$

one dot product $v_o * v_c$ is required, and 5 dot products $-v_k * v_c$ summing from 1 to 5 is required to calculate the negative sampling loss function. Therefore, a total of 6 cross product operations is required.

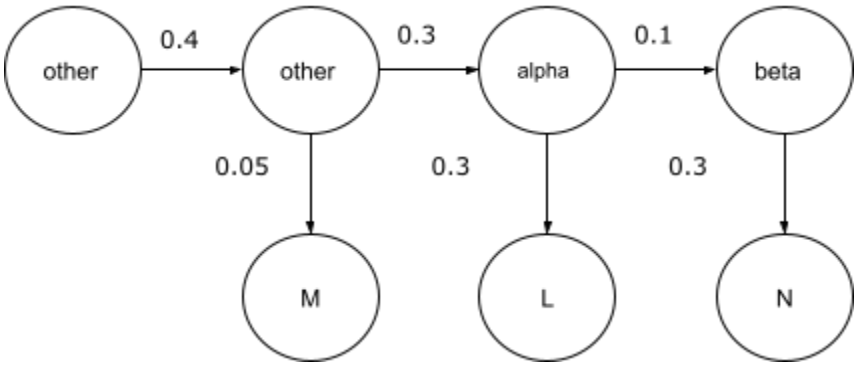
4.1

$P(O, S)$

$= P(O_1 \mid S_1) * P(S_1 \mid S_0) * P(O_2 \mid S_2) * P(S_2 \mid S_1) * P(O_3 \mid S_3) * P(S_3 \mid S_2) * P(O_4 \mid S_4) * P(S_4 \mid S_3) * \ldots * P(O_T \mid S_T) * P(S_T \mid S_{T-1})$

$= A(S_0, S_1) * B(S_1, O_1) * A(S_1, S_2) * B(S_2, O_2) * A(S_2, S_3) * B(S_3, O_3) * A(S_3, S_4) * B(S_4, O_4) * \ldots * A(S_{T-1}, S_T) * B(S_T, O_T)$

4.2



Joint probability = 0.4 * 0.05 * 0.3 * 0.3 * 0.1 * 0.3 = $5.4 * 10^{-5}$

4.3

All possible combinations to generate [M, L, N]:

1. (other, alpha, beta): 0.000054
2. (other, beta, alpha): 0.000009
3. (beta, alpha, other): 0.000072
4. (beta, other, alpha): 0.0000405
5. (alpha, other, beta): 0.0000189
6. (alpha, beta, other): 0.000021
7. (alpha, alpha, alpha): 0.00231525
8. (beta, beta, beta): 0.000162
9. (other, other, other): 0.000096
10. (alpha, alpha, beta): 0.0006615
11. (alpha, beta, alpha): 0.00001575
12. (beta, alpha, alpha): 0.000189
13. (beta, beta, alpha): 0.000027
14. (beta, alpha, beta): 0.000054
15. (alpha, beta, beta): 0.0000945
16. (other, other, beta): 0.000108
17. (other, beta, other): 0.000012
18. (beta, other, other): 0.000072
19. (alpha, alpha, other): 0.000882
20. (alpha, other, alpha): 0.00014175
21. (other, alpha, alpha): 0.000189
22. (beta, beta, other): 0.000036
23. (beta, other, beta): 0.000081
24. (other, beta, beta): 0.000054
25. (other, other, alpha):0.000054
26. (other, alpha, other): 0.000072
27. (alpha, other, other): 0.000252

The total probability is the sum of all the above, which is 0.00578415.

4.4

Assumptions: $o_{1,}$ M is preceded by "others", equivalent to starting token <s> in a sentence.

The calculations below show each entry of the table under the Viterbi algorithm:

	M	L	N
alpha	(1)	(4)	(7)
beta	(2)	(5)	(8)
other	(3)	(6)	(9)

(1) v_1(alpha) = P(alpha | other) * P(M | alpha) = 0.3 * 0.35 = 0.105

(2) v_1(beta) = P(beta | other) * P(M | beta) = 0.3 * 0.1 = 0.03

(3) v_1(other) = P(M | other) = 0.4 * 0.05 = 0.02

(4) v_2(alpha) = P(L | alpha) * max(0.105 * P(alpha | alpha), 0.03 * P(alpha | beta), 0.04 * P(alpha | other)) = 0.3 * max(0.105 * 0.7, 0.03 * 0.2, 0.04 * 0.3) = 0.02205

(5) v_2(beta) = P(L | beta) * max(0.105 * P(beta | alpha), 0.03 * P(beta | beta), 0.04 * P(beta | other)) = 0.05 * max(0.105 * 0.1, 0.03 * 0.6, 0.04 * 0.3) = 0.0009

(6) v_2(other) = P(L|other) * max(0.105 * P(other | alpha), 0.03 * P(other | beta), 0.04 * P(other | other)) = 0.15 * max(0.105 * 0.3, 0.03 * 0.3, 0.04 * 0.04) = 0.004725

(7) v_3(alpha) = P(N | alpha) * max(0.02205 * P(alpha | alpha), 0.009 * P(alpha | beta), 0.004725 * P(alpha | other)) = 0.15 * max(0.02205 * 0.7, 0.009 * 0.2, 0.004725 * 0.3) = 0.00231525

(8) v_3(beta) = P(N | beta) * max(0.02205 * P(beta | alpha), 0.009 * P(beta | beta), 0.004725 * (beta | other)) = 0.3 * max(0.02205 * 0.1, 0.0009 * 0.6, 0.004725 * 0.2) = 0.0006615

(9) v_3(other) = P(N | other) * max(0.02205 * P(other | alpha), 0.009 * P(other | beta), 0.004725 * (other | other)) = 0.2 * max(0.002205 * 0.2, 0.0009 * 0.2, 0.004725 * 0.4) = 0.000882

After filling in the entries and backtrace, the table is as follows:

	M	L	N
alpha	0.105	0.02205	0.00231525
beta	0.003	0.0009	0.0006615
other	0.02	0.004725	0.000882

According to the tracing (in red, tracing for suboptimal paths are excluded), the optimal path is alpha -> alpha -> alpha.