NLP Questions on Language Modelling and Classification (weeks 2 and 3)

Joe Stacey

January 2024

1 Naive Bayes

1.1

State the independence assumption used in Naive Bayes.
Answer: $P(x_1, ..., x_n|y) = P(x_1|y) \times P(x_2|y) \times .... \times P(x_n|y)$

1.2

For a Bag of Words model, provide an example of two features (words) where this is not an accurate assumption.
Answer: Any two highly correlated features (given the class). So any x_1, x_2 where $P(x_1, x_2|y)$ is not equal to $P(x_1|y) \times P(x_2|y)$
For example, in a sentiment analysis task with movie reviews, the words 'mind blowing' are likely to be correlated for the positive class.

1.3

Consider the training corpus in Table 1. Create a Binary Naive Bayes model based only on the features 'good', 'great', 'bad' and 'awful'. Use this model to predict the class of the following review: "tom cruise did it again , so great it's a masterpiece , don't listen to the bad reviews , just enjoy this great film"
Answer: $P(+) = \frac{4}{7}$, $P(-) = \frac{3}{7}$
$P(great|+) = \frac{2}{4}, P(great|-) = \frac{0}{3}$
$P(good|+) = \frac{1}{4}, P(good|-) = \frac{1}{3}$
$P(bad|+) = \frac{1}{4}, P(bad|-) = \frac{1}{3}$
$P(awful|+) = \frac{0}{4}, P(awful|-) = \frac{1}{3}$

We find $P(sentence, +) = P(sentence|+) \times P(+) = P(great|+) \times P(bad|+) \times P(+)$
$= \frac{2}{4} \times \frac{1}{4} \times \frac{4}{7}$

Training corpus (after some pre-processing)	Class
almost as good as the first top gun	+
fun throughout , definitely recommend this great film	+
awful , tom cruise had no depth once again . why is his acting so bad	-
better than i expected , not bad at all	+
lived up to the name top gun , what a great film	+
wish i had just rewatched the original , this one was nowhere near as good	-
tom cruise should do the stunts and leave the acting to someone else	-

Table 1: Film reviews, categorised as being either positive (+) or negative (-)

Whereas $P(sentence, -) = P(sentence|-) \times P(-) = P(great|-) \times P(bad|-) \times P(-)$
$= \frac{0}{3} \times \frac{1}{1} \times \frac{3}{7}$

As $\frac{2}{4} \times \frac{1}{4} \times \frac{4}{7} > 0$, the model predicts the example to have the positive class.

2 Logistic Regression

2.1

What is one limitation of a Naive Bayes model that logistic regression helps to overcome.
Answer: Logistic regression is better at handling highly correlated features, as in logistic regression the model can learn to give a lower weighting to one of the highly correlated features.

2.2

What is the difference between a generative and discriminative algorithm? Is Logistic Regression generative or discriminative? Provide your justification.
Answer: A discriminative algorithm learns which features from the input are most useful to discriminate between the different classes, while a generative algorithm considers the likelihood of the features in an observation given a specific class.

In other words, discriminative models learn $P(y|x)$ directly, while generative algorithms use Bayes rule and find $P(x|y)$.
Logistic regression is therefore discriminative, as it does not use $P(x|y)$, instead directly learning features that help the model distinguish between the different classes.

2.3

Write out the logistic function of an input x.
Answer: $g(x) = \frac{1}{1+e^{-(wx+b)}}$

2.4

How many parameters do we have for a 3-class logistic regression model with 10 different Bag of Words input features?
Answer: For our weights W we have: 3 x 10 = 30 parameters. We also have 3 bias term parameters.

So in total, we have 33.

3 Accuracy and F1-scores

3.1

State the definition of the F1-score for a single class, writing this in terms of precision and recall.

Answer: $\frac{2 \times Precision \times Recall}{Precision + Recall}$

3.2

State the definition for micro-averaged F1, and show how this is equivalent to accuracy. Avoid making any assumptions about the number of classes.

Answer: Micro-averaged F1 is $\sum_i^C \frac{TP_i}{TP_i + \frac{1}{2}(FP_i + FN_i)}$ where C are the different dataset classes.

We can partition our dataset into correct predictions and incorrect predictions. Correct predictions are the $\sum_i^C TP_i$ observations.

Now consider the incorrect predictions: each incorrect prediction must be wrongly predicted as being some class c", and therefore is included in $\sum_i^C FP_i$. As the example also has some true class c', which was not predicted, the example must also be in $\sum_i^C FN_i$.

Therefore, the set of all incorrect observations $= \sum_i^C FP_i = \sum_i^C FN_i$

We can therefore write:
Micro-averaged F1 $= \sum_i^C \frac{TP_i}{|Correct| + \frac{1}{2}(|Incorrect| + |Incorrect|)} =$
$\sum_i^C \frac{TP_i}{|Dataset|}$

This is the same as the definition of accuracy.

3.3

How is macro-averaged F1 different from micro-averaged F1? What are the advantages of using the macro-averaged F1 score?
Answer: The micro-averaged F1 is defined as:

$\sum_i^C \frac{TP_i}{TP_i+\frac{1}{2}(FP_i+FN_i)}$

Whereas the macro-averaged F1 is the mean of the F1 scores for each individual class. The macro-averaged F1 score is therefore more influenced by minority classes, as the F1 scores for each class are averaged (without taking into account the size of each class).

Micro-averaged F1 on the other hand is heavily influenced by the largest classes. For example, if we have 10,000 observations, where 10 are positive and the remainder are negative, then predicting negative for all observations will produce a very high micro-averaged F1 score, despite the poor performance on the positive class.
As a result, macro-averaged F1 is more appropriate for imbalanced datasets.

4 N-grams and perplexity

4.1

Write an expression for approximating $P(w_n|w_1^{n-1})$ using an N-gram model

Answer: $P(w_n|w_1^{n-1}) \approx \mathrm{P}(\mathrm{w}_n|w_{n-N+1}^{n-1})$

4.2

Decompose the joint probability of words $P(w_1, ..., w_n)$ into a product of conditional probabilities after applying a trigram assumption.

Answer: $P(w_1, ..., w_n) \approx \prod_i^n \mathrm{P}(\mathrm{w}_i|w_{i-N+1}^{i-1})$

Note: It is also fine to write:

Answer: $P(w_1, ..., w_n) \approx \prod_i^n \mathrm{P}(\mathrm{w}_i|w_{Max(i-N+1,1)}^{i-1})$

	hamish	is	the	sweetest	cat
hamish	0.01	0.4	0.2	0	0
is	0.02	0	0.4	0.08	0.01
the	0.01	0	0	0.45	0.3
sweetest	0.01	0	0.01	0	0.5
cat	0	0.15	0.03	0	0

Table 2: The column on the left hand side are the words in the history, while the top row are the words being predicted. The values in the table are the model probabilities for the words in the top row, given the words in the left hand side column.

4.3

Given the probability table in Table 2, calculate the probability of the sentence "hamish is the sweetest cat", using a bi-gram model. You can use the information that $P(hamish| < s >) = 0.1$, and $P(< /s > |cat) = 0.1$.

Answer: Probability $= P(hamish| < s >) \times P(is|hamish) \times P(the|is) \times P(sweetest|the) \times P(cat|sweetest) \times P(< /s > |cat)$
$= 0.1 \times 0.4 \times 0.4 \times 0.45 \times 0.5 \times 0.1$
$= \frac{1}{10} \times \frac{4}{10} \times \frac{4}{10} \times \frac{4.5}{10} \times \frac{5}{10} \times \frac{1}{10}$
$= 360 * 10^{-6}$
$= 3.6 * 10^{-4}$

4.4

Would you expect implementing Add-1 smoothing to increase or decrease this probability? Why?
Answer: This question is a bit awkward. The +K smoothing will reduce the probability of the most common bigrams, while increasing the probability of the least common bigrams (e.g. for bigrams where no counts exist). You can see this if you do a simple example, e.g. with a vocabulary of 4 bigrams with counts of 9, 10, 0 and 1. +1 smoothing reduces the likelihood of the first two bigrams (with counts of 9 and 10), while increasing the likelihood of the last two bigrams (with counts of 0 and 1). The test sentence uses the most common bigrams from the table, so we expect the smoothing to decrease the probability, but I didn't give you enough information for you to show this more rigorously (e.g. you don't have total size of vocabulary or counts per bigram).

4.5

What would the perplexity be for this example? Please include $P(< /s > |cat)$ in this calculation.

Answer: ${\frac{1}{360*10^{-6}}}^{\frac{1}{6}}$
$= \frac{10}{360^{\frac{1}{6}}}$
$= \frac{10}{2.67}$
$= 3.75$

4.6

In general, provide an expression for the cross-entropy loss (using log base 2) in terms of perplexity.
Answer: $ppl = 2^H$
Therefore $H = log_2(ppl)$
where ppl describes perplexity, and H is the cross-entropy loss (bits per character)

4.7

Is perplexity an intrinsic or extrinsic measure of performance?
Perplexity is an intrinsic measure of performance.

4.8

Why might it be preferable to evaluate the extrinsic performance of a language model when this is possible?
Answer: An extrinsic measure of performance assesses how well a model performs at the end use task you actually want to assess. Intrinsic evaluation is a less direct way of measuring performance on this end use task, and therefore extrinsic measures of performance are preferable when possible.

5 Feed-forward neural language models

5.1

Consider a feed-forward language model using 4 words of context, 200-dimensional word embeddings and a vocabulary of 20,000 words. Assume no bias terms in the model.

How many parameters are there in the model? Include parameters in the embedding layer.
Answer: Embedding layer (20,000 x 200),
Output layer (4 x 200 x 20,000)
Total: 4,000,000 + 16,000,000 = 20,000,000

5.2

Name two advantages that a feed-forward language model has over an N-gram model.
Answer: 1) The model can make use of pre-trained word embeddings, and 2) the feed-forward language model avoids the sparsity issues that we have with N-gram models

5.3

With a large training corpus, would you expect a feed-forward language model to outperform an n-gram model?
Answer: Yes (you may see 10-20% improvement over a smoothed 3-gram language model).

6 GRUs

6.1

Consider a GRU with:

- 200 dimensional word embeddings
- 100 dimensional hidden states
- 3 output logits (for a multi-class classification task)
- No bias terms in the network
- A maximum of 50 words per observation in the training data
- The GRU is bidirectional (and is a single layer GRU)

Excluding the parameters in the embedding layer, how many model parameters are there in total in the GRU? Please include any parameters in the output layer.

Answer: Consider the equations for a unidirectional GRU (without bias terms):
$g_t = tanh(W_{ig}x_t + r_t * (W_{hg}h_{t-1}))$
$h_t = (1 - z_t) * g_t + z_t * h_{t-1}$
$r_t = \sigma(W_{ir}x_t + W_{hr}h_{t-1})$
$z_t = \sigma(W_{iz}x_t + W_{hz}h_{t-1})$
$y_t = W_y h_t$

Our parameters are: W_{ig} (100×200), W_{hg} (100×100), W_{ir} (100×200), W_{hr} (100×100), W_{iz} (100×200), W_{hz} (100×100) and W_y (100×3).

We therefore have 20,000 + 10,000 + 20,000 + 10,000 + 20,000 + 10,000 + 300 = 90,300

Excluding the parameters in the embedding layer, a bidirectional GRU has twice the parameters of a unidirectional GRU. We therefore have 180,600 parameters.

Note that we did not use the information about there being a maximum of 50 words per observation in the training data. This does not influence the number of model parameters.

7 LSTMs

7.1

Provide the full equations used to define an LSTM (without including bias terms). Do not include an output layer.

Answer (bias terms removed from equations, as specified by the question):

$$i_t = \sigma(W_{ii}x_t + W_{hi}h_{t-1})$$

$$f_t = \sigma(W_{if}x_t + W_{hf}h_{t-1})$$

$$o_t = \sigma(W_{io}x_t + W_{ho}h_{t-1})$$

$$g_t = \tanh(W_{ig}x_t + W_{hg}h_{t-1})$$

$$c_t = f_t * c_{t-1} + i_t * g_t$$

$$h_t = o_t * \tanh(c_t)$$

7.2

State the dimensions for each matrix and vector mentioned in the equations above, where we have E dimensional word embeddings and H dimensional hidden states.

Answer (bias terms removed from equations, as specified by the question): The dimensions of each matrix are: W_{ii} (HxE), W_{hi} (HxH), W_{if} (HxE), W_{hf} (HxH), W_{io} (HxE), W_{ho} (HxH), W_{ig} (HxE) and W_{hg} (HxH)

The dimensions of each vector are: x_t (Ex1), h_t (Hx1), c_t (Hx1), f_t (Hx1), g_t (Hx1), o_t (Hx1) and i_t (Hx1)

7.3

Explain why LSTMs can help to mitigate the vanishing gradient problem seen in vanilla-RNNs, mentioning the role of the forget gate.

Answer: In vanilla-RNNs, when backpropagating through time, finding the gradient of one hidden state with respect to another hidden state from a previous time-step involves repeated multiplication by the matrix W in our vanilla RNN. This repeated multiplication by W can either cause gradients to vanish or explode, depending on the value of the dominant eigenvalue in the matrix W.

On the other hand, in LSTMs information from long range dependencies across the sentence can be stored in c_t, which does not have the same vanishing gradient issue. More specifically, when finding the gradients of c_t with respect to c_{t-k} for some k, we do not have repeated multiplication of the same matrix W. Instead, the gradient of c_t depends on the forget gate (f_t), where the forget gate takes different values for each time-step (depending on the weights in the forget gate learnt by the model). The model can therefore learn when the gradients for c_t should vanish (with a lower value of f_t), or when the gradient should be preserved.

Note: it is possible to go into more detail about why LSTMs help to mitigate vanishing gradients, but this is the level of detail we have covered in the course.

7.4

One option for a sentence-level classification task is to apply a classifier to the final hidden state of an LSTM. Suggest two different strategies we could use instead of using the final LSTM hidden state.
Answer: We could 1) mean pool or 2) max pool across the hidden states at each time step before we apply our classifier.

8 True or False questions

8.1

Binary classification models can either have a single logit output (using a sigmoid), or they can have two logit outputs (using a softmax). Both methods are acceptable.
Answer: True! Both are acceptable methods.

8.2

A non-linear activation function usually precedes softmax in a model.
Answer: False. A softmax expects an unbounded input, so this is usually applied after the final linear layer.

8.3

Prior to transformer and RNN models, neural networks in NLP usually had more than 5 layers.
Answer: False. Prior to RNNs and transformer models, 2-3 layers were usually enough for neural networks in NLP.

8.4

De-biasing methods usually have no impact on a model's performance on in-domain test sets.
Answer: False. De-biasing methods usually reduce performance on in-domain test sets.

8.5

Excluding the embedding layer, a single-layered bi-directional LSTM has exactly twice the number of parameters of a single-layered unidirectional LSTM.
Answer: True! Even the output layer has twice the number of parameters (can you see how?)