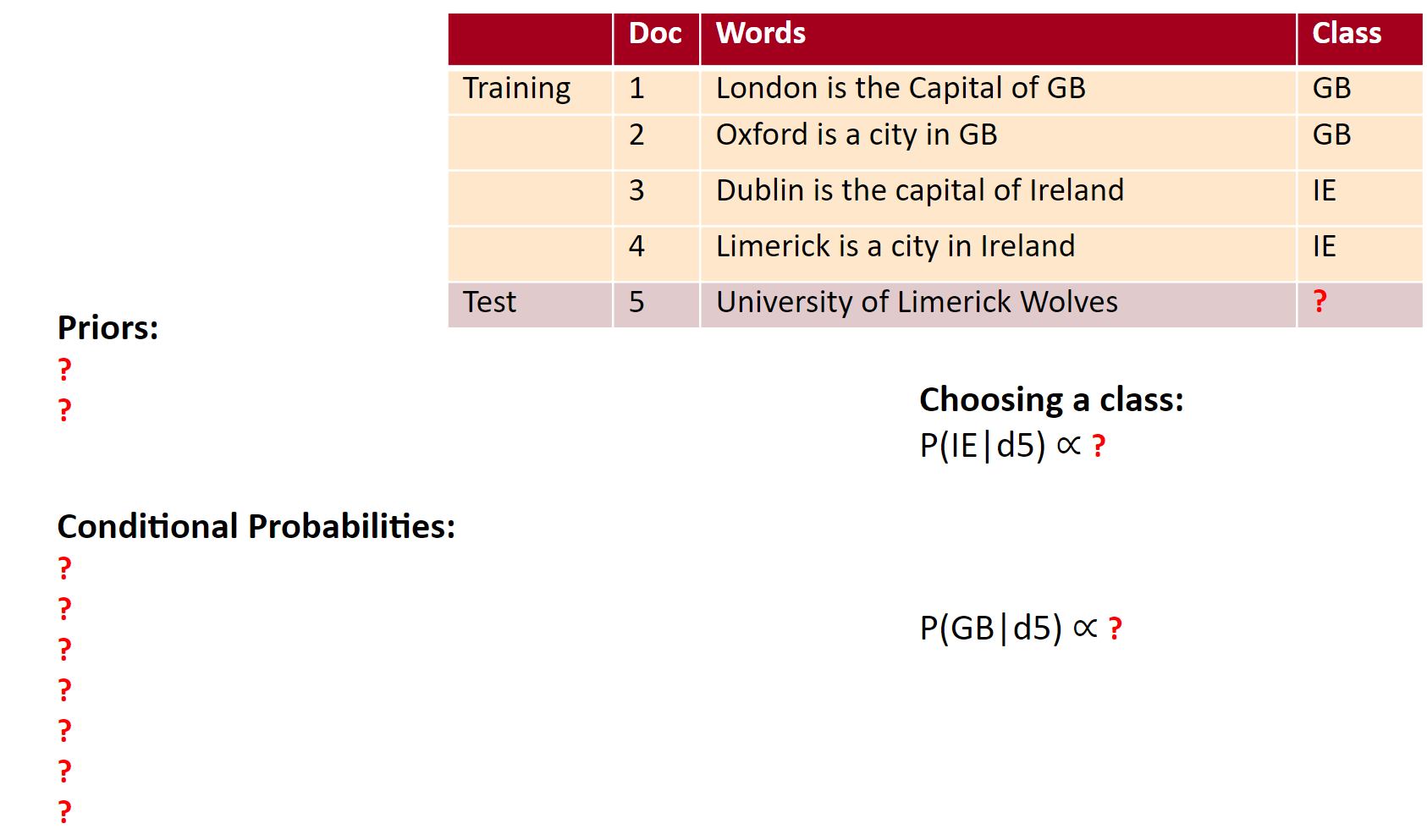
# Task 1: [2 marks]

Given the four training documents in the table below, infer the class of the test document (doc #5). Show your work using Latex, i.e. computations of priors and conditional probabilities.



# [Intro to LaTeX: Learn to write beautiful math equations](https://youtu.be/Jp0lPj2-DQA)

# [Adding Latex to your notebook](https://colab.research.google.com/github/bebi103a/bebi103a.github.io/blob/master/lessons/00/intro_to_latex.ipynb)

# 

# Task 2: [3 marks]

# 

Sentiment analysis is the interpretation and classification of emotions within text. Write a simple rule-based sentiment analyzer, which takes a string as input and returns its sentiment (positive, negative, neutral).

Here’s a basic example of how a rule-based system works:

1. Define two lists of polarized words (e.g. negative words such as *bad*, *worst*, *ugly*, and positive words such as *good*, *best*, *beautiful*).
   * HINT: you can use [Bing Liu's lists of positive and negative sentiment words](https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html#lexicon) ([RAR archive](http://www.cs.uic.edu/~liub/FBS/opinion-lexicon-English.rar)). **You should directly** [***wget***](https://www.computerhope.com/unix/wget.htm) **them to your notebook from:** <http://ptrckprry.com/course/ssd/data/positive-words.txt> and <http://ptrckprry.com/course/ssd/data/negative-words.txt>
2. Count the number of positive and negative words that appear in a given text.
3. If the number of positive word appearances is greater than the number of negative word appearances, the system returns a positive sentiment and vice versa. If the numbers are even, the system will return a neutral sentiment.

## Sample Output: (you need to devise a method for generating confidence scores; you can achieve this in a single line of code)



# Task 3: [5 marks]

Write a NaiveBayes function which takes the sample training and testing documents shown in the table below as input, and uses the Naive Bayes algorithm to classify the test documents.

* Hint1: do not forget to normalise (case-fold) your train & test docs.
* Hint2: [Python Dictionaries](https://www.w3schools.com/python/python_dictionaries.asp) are suitable data structures for storing BOWs (Bag Of Words)

|  | **Doc** | **Words** | **Class** |
| --- | --- | --- | --- |
| Training | 1 | London is the Capital of GB | **GB** |
| 2 | Oxford is a city in GB | **GB** |
| 3 | Dublin is the capital of Ireland | **IE** |
| 4 | Limerick is a city in Ireland | **IE** |
| Test | 5 | University of Limerick | **?** |
| 6 | University College Dublin | **?** |
| 7 | Imperial College London | **?** |
| 8 | University of Oxford | **?** |
| 9 | Ireland & GB | **?** |

## 

## Sample Output:

## 

~~~~

## </END OF ETIVITY>

TODO (for lecturer):

* Level of difficulty (easy/fair/hard)
* Task1: print the total number of positive and negative words
* Stop word removal, stemming, unknown words
* 3 classes
* Task 3: [5 marks] sentiment analysis
* HIDE the inferred class
* Hide the confidence value
* There is the [chardet](http://pypi.python.org/pypi/chardet) library that uses that study to try to detect encoding. chardet is a port of the auto-detection code in Mozilla. <https://stackoverflow.com/questions/436220/how-to-determine-the-encoding-of-text>

## </END OF ETIVITY>

For lecturer:

* Level of difficulty (easy/fair/hard)
* Task1: print the total number of positive and negative words
* Stop word removal, stemming, unknown words
* 3 classes
* Task 3: [5 marks] sentiment analysis
* HIDE the inferred class
* There is the [chardet](http://pypi.python.org/pypi/chardet) library that uses that study to try to detect encoding. chardet is a port of the auto-detection code in Mozilla. <https://stackoverflow.com/questions/436220/how-to-determine-the-encoding-of-text>