**Data Reply Data Science test**

Q1.

1. What are the probabilities of each die generating this data

Each cell contains the probability of a number/event occurring in corresponding dice

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| data\k | 4 sided | 6 sided | 7 sided | 8 sided | 10 sided | 20 sided |
| 5 | 0 | 1/6 | 1/7 | 1/8 | 1/10 | 1/20 |
| 2 | 1/4 | 1/6 | 1/7 | 1/8 | 1/10 | 1/20 |
| 7 | 0 | 0 | 1/7 | 1/8 | 1/10 | 1/20 |
| 4 | 1/4 | 1/6 | 1/7 | 1/8 | 1/10 | 1/20 |
| 2 | 1/4 | 1/6 | 1/7 | 1/8 | 1/10 | 1/20 |
| 6 | 0 | 1/6 | 1/7 | 1/8 | 1/10 | 1/20 |
| 7 | 0 | 0 | 1/7 | 1/8 | 1/10 | 1/20 |

From the above able table, we can interpret as following:-

P(5/4sided) = 0, P(2/4sided) = ¼ and so on

P(5/6sided) = 1/6, P(2/6sided) = 1/6 and so on

……. till

P(5/20sided) = 1/20, P(2/20sided) = 1/20 and so on

Also since there are 6 dice and so selecting random makes the probability of picking any of the dice P(ksided) = 1/6 = P(4sided) = P(6sided) = …

According to

P(4sided/d) =

(P(d/4sided) \* P(4sided))/P(d) and assumption taken as P(d) = 1

Where pattern d contains following – 5 ,2 ,7, 4, 2, 6, 7 and P(4sided) = 1/6

= (P(5/4sided)\*P(2/4sided)\*P(7/4sided)…P(7/4sided) \* 1/6

= 0 \* ¼ \* 0 \* ¼ \* ¼ \* 0 \* 0 \* 1/6 = 0

P(6sided/d) =

……

1. On average, how many rolls would it take until one could effectively rule out the 20-sided die

Let’s say number of rolls = x

Probability of not getting 20 till x -1 rolls = (19/20)**x-1** which should be = ½ for average case

Equating the above to into following statement and applying log on both sides

(19/20)**x-1** = ½

Take log on both sides (x-1)\*log(19/20) = log(1/2) = log1 – log2

Solving these x = [-1 + log2(5/6)]/log2(5/6)

1. 3 consecutive 1s

Let’s say all numbers till k except 1 = (k-1)/k,

Then 1 followed by any of the numbers except 1 = (k-1)/ k2

Then 1 followed by any of the numbers except 1 = 1/ k2

#TBD

1. TBD

Q2. Word Count

E.g. below n lines of text:-

I am working on assignment

I am finding this assignment difficult

….

I hope I should be able to crack it in few hrs

Some common better solutions can be used.

Most fundamental and easiest 2 implement is the use of For eg in python is using a dictionary which in JAVA similar to HashMap to maintain the word as key and occurance as the value of the word

Assumption is that file is too big to handle for my local desktop and so Hadoop environment can be setup on the server

Here is the algorithm:-

1. Input the file
2. Break the fil into n approx. equal chunks
3. These broken parts can be transferred over x number of machines on the cloud where Hadoop is setup.
4. HDFS ingest the broken pieces by keeping parts of files on different servers
5. Now with the help of MapReduce algorithm we can feed in the following steps to enable distributed computing
6. **Map step**:- It gets a word. Emit the word followed by 1 as the key value pair. Keep on doing it until we have read all the words from the chunk.
7. **Reduce step**:- here it takes the key value pairs as input. It initializes an empty word dictionary just like a HashMap. It checks if the word it is reading is the same as if already present in its key value dictionary. If yes , then it increments the value of occurrence of the word in the dictionary by 1 else adds the word in the dictionary
8. Finally Output in word dictionary is written to external file as per the user requirement for analysis

In terms of tradeoff in speed and memory, with the help of Hadoop we shall be able to solve if the file being talked about is in 100s GB other. Data transfer over the network will also be need to be looked at while implementing this algorithm.

Above algorithm/business logic if performed on a single computer which is mainly in **Reduce step** have time complexity as O(n).

Handling misspelled words in word count:-

It depends on the ROI as each solution requires a certain amount of investment in terms of time resources and infrastructure

1. Assuming the words are misspelled are not the ones from stop words list and have significance in subsequent predictive modeling tasks.

A Language model with already labelled data can be used to identify whether a particular word is misspelled and eventually replaced with the correct word

For eg <http://ota.ox.ac.uk/headers/0643.xml> assuming the text is in English

1. Other option would be build the complete language corpus/dictionary of misspelled words and with the manual crowdsource effort of labelling it with correct words.Accuracy will be keep on increasing as we find more misspelled words reported and added to the dictionary.
2. Also statistical methods such as Levenshtein distance between words can be used as metric into the model to boosten the predictability of the word as misspelled or not.
3. Techniques like RNNs can be used to correct the spelling as shown here <http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-2-implementing-a-language-model-rnn-with-python-numpy-and-theano/>

Q3.

Common words in the LinkedLIst

1. Lets say head1 = head of linkedlist 1
2. head2 = head of LinkedList2
3. Iterate through each element of list1
4. If the element is present in list2 then save the element in result 2 if it is not already present there

This is a brute force algorithm which will be feasible if the linkedlist size are small i.e. in 1000 elements

Other is divide and conquer algorithm:-

1. Sort the both Linkedlists using mergesort
2. Initialize head1 and head2
3. Start iterating over the linkedlist together with each having the pointers as pointer1 and pointer 2 as respectively
4. If data(pointer1)< data(pointer2) then pointer1 = pointer1.next i.e. move the pointer1 to the next element in Linkedlist1
5. If data(pointer1) > data(pointer2) then pointer2 = pointer2.next i.e. move the pointer2 to the next element in Linkedlist2
6. If data(pointer1) == data(pointer2) then add the data to the result and increment both pointers i.e. pointer1 = pointer1.next and pointer2 = pointer2.next

This algorithm has complexity as O(m+n) excluding the sorting and sorting of the linkedlist steps have the complexity as O(nlogn) + O(mlogm)

Q B4

Generative algorithm that can be used according to given dataset here is Gaussian Discriminant Analysis or naïve Bayes and Discriminant algorithms technique which can be used here is SVM or logistic regression

Discriminative models learn the (hard or soft) boundary between classes For eg SVM, decision trees, logistic regression. It simply doesn’t the care about the data distribution. These work better on a large training size.These are more like blackbox as relationships between variables are not explicit and visualizable

Generative models model the distribution of individual classes – Naïve Bayes. They model the conditioning probabilities of the classes. Then based on the decision thory it can be determined what data point belongs to which class using loss matrix. It cares about the data distribution. These perform better on less training data. These are explanatory models

Performance of the classifier can be assessed by statistics by drawing confusion matrix and chunking out statistics such as classification accuracy, sensitivity(Recall), specificity, Precision(positive predicted value). Also checking the model whether any imbalance in the dataset. KS statistic can be used for this.

Optimize the model can be done using like ensembling i.e. Combining multiple classifiers and giving a combined output either by averaging or choosing the majority rules etc. Also algorithm tuning can be done by tuning techniques like Cross Validation i.e. breaking a training data into multiple parts and do validation in multiple rounds taking one of them as validation set and rest as training. Rest is from domain expertise, is doing more feature engineering so as to extract better features.

To describe unlabeled data, clustering techniques can be employed. For eg if it is a text data then topic modeling techniques e.g. work very well in identify topics which can be used to label the data. For numerical data techniques like k-means work very well in identifying the clusters.

To assess the quality of clustering assignments metrics like Adjusted Rand Index, homogeneity scores, completeness. Silhoutte Coefficient is one of the very popular metrics to evaluate the performance of cluster assignments. Usage of a particular metric is bases on the distribution of the data. Calinski-Harabaz index here will fit on the images shared as the clusters are seemly dense and well separated. All can be found here

<http://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation>

Puneet Jindal

+91-783-886-0769

Puneet\_jindal\_2014@cba.isb.edu