1. What is the asymptotic (big O) runtime complexity of the methods: setTraining() getRandomText() for the BaseMarkov impelementation in terms of N and T?

setTraining() - O(T)

This method calls the split() method once. Therefore the big-O runtime complexity is O(T).

getRandomText() - O(NT)

This method initializes an arraylist once which is constant time complexity. Then it generates a random integer once which is constant time complexity. Then it creates a wordgram object once which is constant time complexity. Then it calls toString() on a wordgram object once which is constant time complexity with respect to N and T. Then it adds a string to an ArrayList which is constant time complexity. Then a forloop runs N times. Inside it, getNext() runs N times which has linear time complexity O(T), because it calls getFollows(). add() runs N times which has constant time complexity in respect to N and T. shiftAdd() runs N times which has constant time complexity in respect to N and T. Thus the total runtime complexity is O(NT).

Data file	T	N	Training time (s)	Generating time (s)
alice.txt	28196	100	0.016	0.146
alice.txt	28196	200	0.016	0.240
alice.txt	28196	400	0.017	0.394
kjv10.txt	823135	100	0.162	2.187
kjv10.txt	823135	200	0.159	4.026
kjv10.txt	823135	400	0.191	8.059
shakespeare.txt	902325	100	0.114	2.387
shakespeare.txt	901325	200	0.130	4.176
shakespeare.txt	901325	400	0.112	8.394

Looking at this data, we can see that as N doubles, the generating time doubles as well. This is in line with the linear time complexity with respect to N theory. As T increases ~30x between alice.txt and kjv10.txt, the generating time also increases by a relatively close amount. Also between shakespeare.txt and kjv10.txt which have a similar amount of words, the generating time is similar. This is in line with the linear complexity with respect to T theory.

2. What is the asymptotic (big O) runtime complexity of the methods: setTraining() getRandomText() for the HashMarkov implementation in terms of N and T?

setTraining() - O(T)

This method calls the String split() method once which has linear time complexity with respect to T. Then the method calls the hashmap clear() method which has linear time complexity with respect to the size of the map, but in this experiment the method never gets called, because the map is never full before running this method. Then there is a forloop which runs T times. Inside the loop, an ArrayList is constructed, a wordgram is constructed, the hashmap put() method is called, the hashmap get() method

is called, and the hashmap add() method is called. All of these computations inside the loop have linear time complexity and run T times. Thus the method has big-O time complexity O(T).

getRandomText() - O(N)

The method constructs an ArrayList once which is constant time complexity. Then it generates a random integer once which is constant time complexity. Then it creates a wordgram object once which is constant time complexity. Then it calls the arraylist add() method once which is amortized constant time complexity. Then a forloop runs N times. Inside it, getNext() runs N times which has constant time complexity, because getFollows() is constant, as opposed to being linear time complexity in BaseMarkov. This makes HashMarkov much more efficient for large N. add() runs N times which has constant time complexity in respect to N and T. shiftAdd() runs N times which has constant time complexity in respect to N and T. Thus the expected total runtime complexity is O(N).

Data file	T	N	Training time (s)	Generating time (s)
alice.txt	28196	100	0.052	0.001
alice.txt	28196	200	0.055	0.002
alice.txt	28196	400	0.053	0.003
kjv10.txt	823135	100	0.593	0.003
kjv10.txt	823135	200	0.564	0.003
kjv10.txt	823135	400	0.514	0.005
shakespeare.txt	902325	100	0.620	0.004
shakespeare.txt	901325	200	0.623	0.003
shakespeare.txt	901325	400	0.624	0.006

Looking at this data, we can see that as N increases, the generating time increases as well by a similar factor. This is in line with the linear time complexity with respect to N theory. As T increases ~30x between alice.txt and kjv10.txt, the Training time also increases by a relatively close amount. Also between shakespeare.txt and kjv10.txt which have a similar amount of words, the training time is similar. This is in line with the linear complexity with respect to T theory.

3. What do you think of OpenAI's stated mission? In particular, do you think that "highly autonomous systems that outperform humans at most economically valuable work" can benefit all of humanity? Why or why not?

I think that is a good goal that will lead to good things for humanity. OpenAI is one of the most influential companies for the future of human technology, and with that comes a lot of power. The fact that their mission is to benefit humanity rather than generate profit for shareholders like a public company is comforting. I'm almost certain that autonomous systems will outperform humans in most economically valuable work eventually, and there are outcomes where this is a good thing. If the AI used in these systems is harnessed and trained properly, it will solve many of the problems people face today.