Question 2.1:

Let's start with the latent semantic analysis and the probabilistic latent semantic analysis. In both techniques the main point is to model the underlying topics in the document corpus. In latent semantic analysis this is done by first creating a term-frequency matrix and then approximating it by the singular value decomposition, which in practice is a dimensionality reduction method. The singular value decomposition allows collecting the most informative data from the term frequency matrix which corresponds to detecting the most prominent underlying topics of the document related to terms. In the other case, probabilistic latent semantic analysis focuses on detecting the co-occurrences of words and documents. This is done by assuming that there exist a set of underlying topics, that are present but not known (only the number of the topics has to be defined). Like the name says, this is a probabilistic technique (LSA is mainly based on linear algebra only) and the joint probability of the word and document is decomposed as a sum over possible multiplications of topic prior probabilities and conditional word & document probabilities given the topic. Latent Dirichlet Allocation focuses is also a probabilistic model as PLSA, but it assumes that each document has its own mix of topics and each topic has its own mix of words.

Question 2.2:

In general, the HMM is more simpler generative model compared to the LSTM but they also have somthing in common. They both are intended to model time sequence models and probabilities related to them. In HMMs the time sequence is modelled via latent (hidden) and observable states, where the transition probabilities between the latent states and the emission probabilities of the observable states given the latent state have to be estimated to do any meaningful analysis. When these probabilities are estimated, it is possible to generate a time sequence of some type (eg. the sequence of words) pretty easily, by sampling the sequence of hidden states and generating observable values (eg. words) from those hidden states. Learning of the parameters of HMM is usually accomplished by dynamic programming techniques, eg. Forward-Bckward algorithm, Viterbi algorithm or Baum-Welch algorithm. LSTM, in the other case, is a neural language model highly related to deep learning. It is based on the general neural networks, but the computing nodes are more complicated and they can model a simple memory-cell structure inside of the computing node by feedback connections. This memory structure accomplish to take previous time steps into account (like the HMMs, where the next time step is dependent to the previous one) when calculating the output with current values. Learning of the LSTM is done by gradient descent algorithm.

Question 2.3:

Given probabilities are not possible. This can be noted if we write out the probability of $p(w_1 = rock|\ w_2 = band)$

$$\begin{aligned} p(w_1 = \text{'rock'}|\ w_2 = \text{'band'}) &= \frac{p(w_2 = \text{'band'}|\ w_1 = \text{'rock'})p(w_1 = \text{'rock'})}{p(w_2 = \text{'band'})} \\ &= \frac{0.4 * 0.01}{0.003} \\ &= \frac{4}{3} \end{aligned}$$

This has to be less than one. Therefore they are not possible.

Question 2.4:

In general, learning a long n-gram model (for example n=10) overfits very easily, since the probabilities of next word is calculated in context of n-1 previous words (9 words). In n-gram models, the conditional probabilities of the following words are calculated from training data (existing document in the corpus). Therefore it is highly likely that in case of a long n-gram, there is only one occurrence for a sequence of previous words and thus the conditional probability of the following word is 1 for one specific word. This behaviour causes the algorithm to only memorize the contents of the training data (the contents of the document).