Implementing these probability calculations, has some practical issues one of which is the phenomena known as *underflow* and *overflow*. Underflow and overflow happen when the numbers become too small or too large to be represented by the datatypes in the programming language and the computer. These are common occurrences in Bayesian Statistics and therefore have well known resolutions. Doing the calculations in the log space is a common solution for overflow and underflow.

This equation is equation ? moved to log space. However, this summation must be turned into log space, since its calculation is the reason we're moving to log space in the first place. There is a common method called log-sum-exp trick used in the context of forward algorithm in HMMs that is applicable here:

It runs based on the idea that is an approximation of the maximum function (). If we factor out the biggest contributor of the sigma (*A*), we can avoid the overflow problem in calculating this sum. This is basically shifting the biggest contributor to zero by doing and then shifting it back by adding *A*. Here .

A similar approach was used when normalizing this joint probability to calculate the conditional probability.

**The likelihood**

The middle term (), henceforth referred to the Log Likelihood (), will be calculated based on Equation ?:

And the initial values are:

**Vector Size Normalization**

The feature vectors that are used in the likelihood calculations might have different sizes because the size of the documents, and therefore the number of words or part-of-speech tags in them, can vary from timestep to timestep. Even if the vector sizes where the same, because we aggregate the documents from previous steps in each node’s likelihood calculations, the higher the node is in each column, the lower its likelihood gets. Comparing feature vectors of different sizes will skew the comparison as it will make the document length a contributing factor. In reality documents that have the same topic should have the same likelihood regardless of their size, therefore, a normalisation step must be employed before feature vectors are compared.

If this algorithm was running offline, we could have simply taken the size of the biggest feature vector from the whole dataset and normalized all other vectors to be of that size, however, our system is online and the sizes of the future data is unknown, thus another approach should be taken.

By exploiting the fact that we are only comparing the likelihoods of the nodes in each timestep together, we can assume different normalised sizes for different timesteps. We set this normalised size to be the running average size of the documents so it does not need to be specified in advanced and may change with every new datum the algorithm receives. To normalise the size, all elements of the feature vector are multiplied by the ratio of the running average and current vector size.

A similar approach is used in determining the size of dictionary (what is dictionary?). As new datum becomes available size of the dictionary may or may not increase.

**Explaining the intuition of the likelihood model**

The likelihood is the probability that illustrates how likely it is for the new datum to belong to the current segment of the time series. If we represent data using the bag-of-words model (ignoring the order in which words appear in the sentence), the unigram language model consists of a set of probabilities for each of the possible words in the language. These probabilities are between zero and one and should add up to one. This gives an unlimited number of possible language models that are bounded by these conditions. In a three-dimensional world (where the language only has three words in its vocabulary), all these language models can be presented by a hyper-plane:

We can use the maximum likelihood principle to find the optimal based on the data available so far, but instead we try to find the most likely language model this datum belongs to. The hyper-plane of all possible language models can be represented using a Dirichlet distribution that has the hyper-plane as its probability simplex.

**Authorship Attribution Task**

Different features were investigated for this task.

POS tag frequency count is one that was not successful in changepoint detection.

Another common feature used for authorship attribution is the the frequency count of function words. Function words are ??.

**Algorithm**

1. Initialisation

P (r0 = 0) = 1

1. For all data xt
   1. For rt = 0 (changepoint probability)
      1. Calculate likelihood with the current data and
      2. Log (joint probability for this point) = Sum over all the points in the previous time step:
         1. Add the log(joint probability) + log(likelihood) + log(prior probability of change)
   2. For all the other possible rt (growth probabilities)
      1. Update the sufficient statistics
      2. Calculating likelihood with the current data and updated sufficient statistics
      3. Log (joint probability for this point) = Joint of the previous point + log(likelihood) + log(prior probability of not having change)
2. Get the point with maximum joint probability at each time step