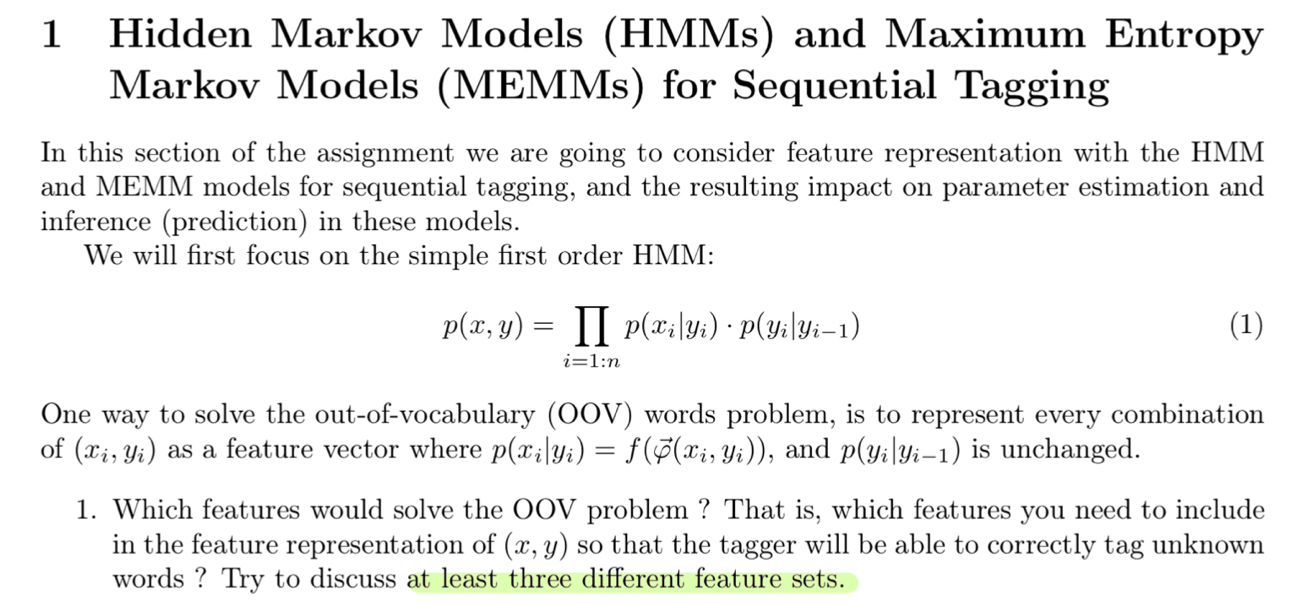
**HW1 NLP - Dry**

Yonathan Mizrahi – 209948728

Raphaël Bellahsen – 931188684

**Question 1:**

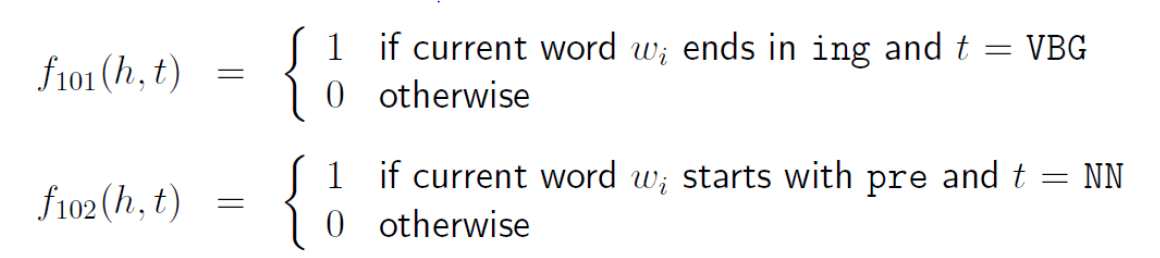


**Answer:**

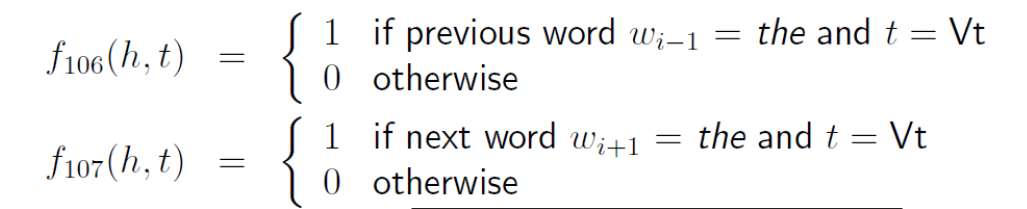
First let’s define the OOV problem: the OOV (out-of-vocabulary) problem are terms that are not part of the ‘normal’ lexicon found in a NLP environement.

There are several features that need to be included to solve the OOV problem:

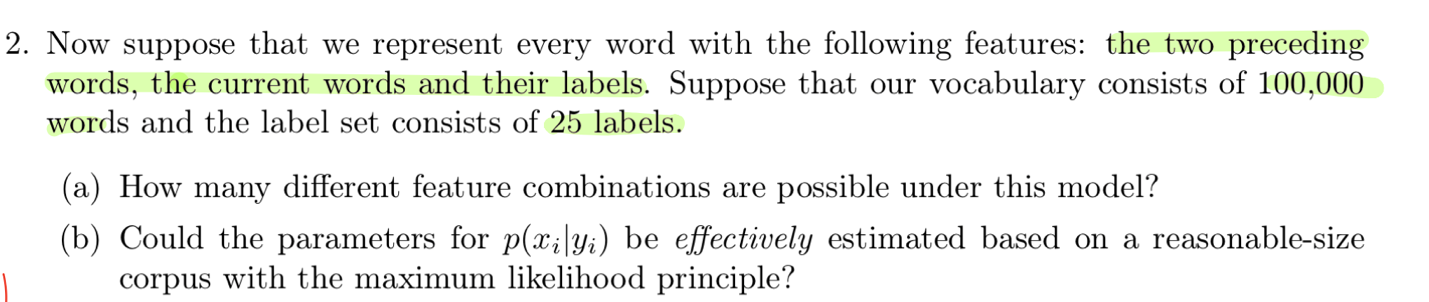
* Features for prefixes/suffixes, like:



* Contextual features based on previous/next word and tag of the current word:



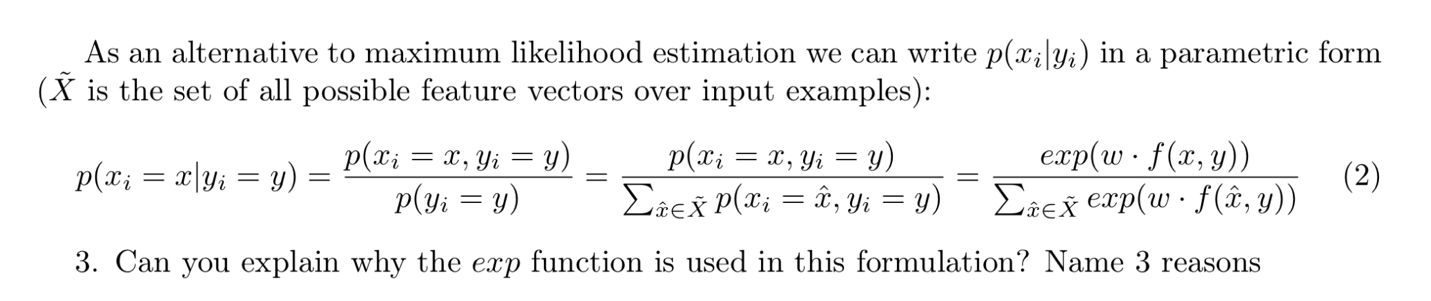
**Question 2:**



**Answer:**

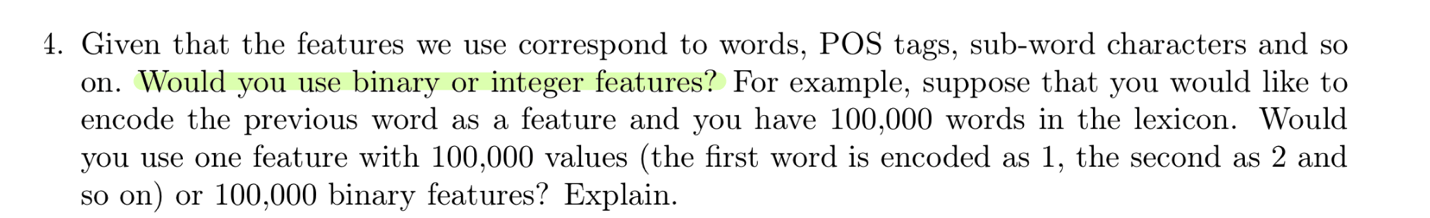
1. In this context, there are 100,000 different possibilities for each word, 25 for each label, so at total: different features.
2. No for this case because there are too many features and it seems that the parameters won’t be effectively estimated.   
   Also, a lot of these features are useless. For example, the probability to see 3 times in a row the same word is near 0, so that feature can be removed.

**Question 3:**



**Answer:**

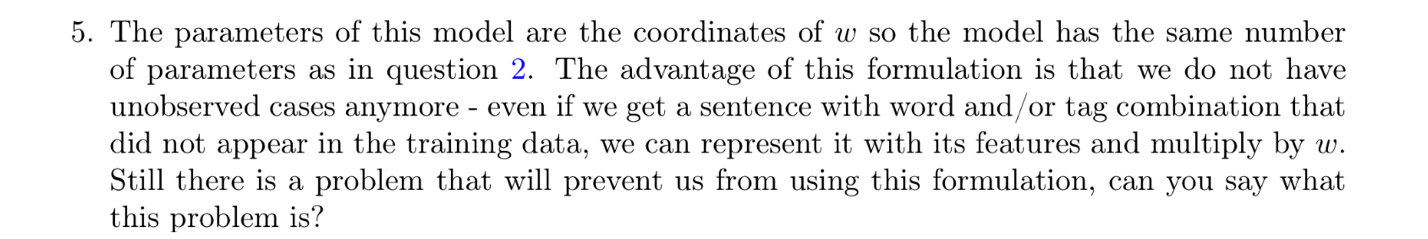
1. The exp function is used to receive only positive numbers. It is important to recall that in writing this formula, we want the result to be a number between 0 and 1. Using exp we transplant all the numbers to be positive. Then dividing by the sum of exp functions the result is as expected, between 0 and 1.
2. Recall that the exp function is monotonically increasing. It’s ensuring that larger inputs can be mapped to larger outputs.
3. When applying log on the probability, we get the log-linear form (as we saw in the lecture). This form is much more convenient to work with since it’s very useful to optimize (easier to differentiate).

**Question 4:**

**Answer:**

We would use 100,000 binary features. By using the integer feature, all our information will be summarized into a single value – our integer. However, by using the binary feature, we let the model learn the weight of each feature and it led to a more diverse representation (which is better).

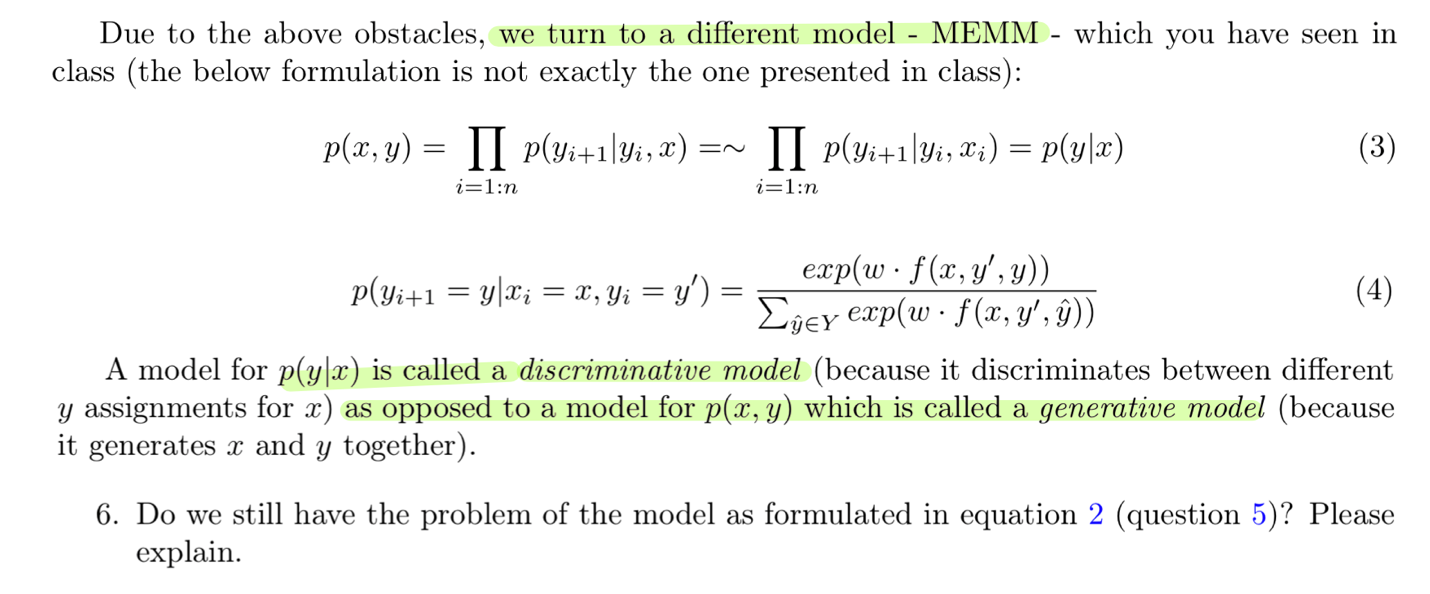
**Question 5:**



**Answer:**

As the formula is written in this section, the model takes a lot of time to train since the calculation has multiples combination to calculate. Indeed, a lot of word combinations must be checked for each label sequence. Moreover, the target and the prediction function do not match, because this model acquires only the joint distribution instead of the conditional probability. Finally, since the formula is likelihood-based, the HMM, cannot condition on any useful feature of the input observation.

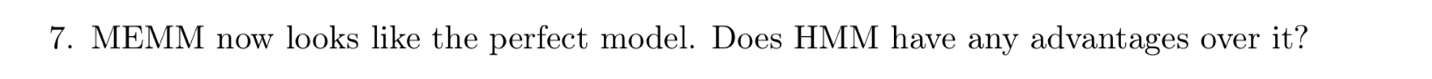
**Question 6:**



**Answer:**

**I**n this discriminative model, we are done with the problem of the generative model (explained just before). Now, we’re computing the conditional probability. Furthermore, the training will be faster, as estimating the parameters can be done for each transition distribution in isolation.

**Question 7:**



**Answer:**

The HMM model is simpler and more comprehensible since we’re directly computing the prob for each class. For that reason, it’s easier to analyze if something (and what) went wrong in the HMM model and then easier to debug (compared to the MEMM model).

Secondly, the HMM model is a generative mode, whereas the MEMM model is a discriminative mode. That means that the HMM model evaluates the probability for each possible class in the problem, whereas the MEMM model estimates the boundaries that differentiate between the possible classes. For some tasks, it would be better to use the generative model, like creating sentences.