True/False (4 pts, recommended 20 minutes)

For each of the statements below, indicate if it is true or false. **If it is false, explain why it is false and provide an example. You may provide an explanation if it is true in case you are wrong and would like to receive partial credit.**

A. The regex **\bThor\b** will result in a higher false positive rate than **Thor** when searching for references to the movie character Thor.

*False. The word boundaries around Thor will cause it to have a lower false positive rate. A false positive here would be matching the word **Thorough**, or **Thorn**, since you are actually looking to match for references to the character Thor. If you use word boundaries, these matches will not happen.*

B. In a Hidden Markov Model's **emission matrix**, assuming rows are **observed states** and columns are **hidden states**, the sum of each row should equal 1.

False, but I won't ask about HMMs in the exam since we did not cover yet. However, the columns, not the rows, of the emission matrix should sum to 1.

C. A document that has original text **cat litter** and another document that has the text **litter cat** will have identical vectors when using word count vectorization, TF-IDF vectorization, and bag-of-words word2vec vectorization.

True. In all of these cases, we are using bag of words, which means as long as the frequency of words is the same, the documents are the same.

For example, let's say the first feature is cat, and the second feature is litter.

Document A is **cat litter**
Document B is **litter cat**

Both would be vectorized as [1, 1].

Now let's pretend that Document A and B are the only two documents in the corpus.

Since they have the same term-frequencies, they'll also have the same TF-IDF scores.

*Now let's pretend that the word2vec embedding for cat **is [0.2, 0.3, 0.4, 0.5, -1.2]** and the word embedding for litter is **[0.3, -0.1, -0.1, -0.3, 1.1].***

*The bag of words document vector using word2vec would be the average of **cat** and **litter** for Document A, and the average of **litter** and **cat** for Document B. Both would be the same document vector.*

1.

D. Two documents:

   a. **cat cat dog dog love love**
   b. **cat dog love**

   Would show a **cosine distance** > 0.

False (assuming a count vectorization technique). This should show a cosine similarity of 1, or a cosine distance of 0.

Let's say the first feature is cat, the second feature is dog, the third feature is love. Then

Document A would be count vectorized as [2, 2, 2].
Document B would be ount vectorized as [1, 1, 1].

The dot product is 2x1 + 2x1 + 2x1 = 6.

The norm of Document A is $\sqrt{2\ x\ 2\ +\ 2\ x\ 2\ +\ 2\ x\ 2}$ *= 3.464*
The norm of Document B is $\sqrt{1\ x\ 1\ +\ 1\ x\ 1\ +\ 1\ x\ 1}$ *= 1.732*
The cosine similarity of Document A and document B is $\frac{6}{3.464\ x\ 1.732} = 1.$
The cosine distance is 1 – 1 = 0. Hence, statement is false.

Note – I didn't explicitly go over cosine distance, so I would not have asked this on this year's exam.

E. There are 3 capture groups in the following regex:
   (?:Mr\.|Miss)\s(\w)\s(?P<last_name>\w)

False – ***(?:Mr\.|Miss)*** *is not a capture group. It is a non-capture group.*

The capture groups in this expression are

1. ***(\w)***
2. ***(?P<last_name>\w)***

The last is a named capture group – I did not cover yet in class, so I would not have asked this on the exam.

F. **UTF-8** and **ASCII** use the same **Unicode codepoint** for the character "a".

2.

True. The code point is 97. ASCII goes from 1-128 codepoints. UTF-8 is a superset of ASCII, so it includes all the code points in ASCII, including the codepoint for "a".

G. If a model's F1 score is 1, it is guaranteed to have 0 false positives and 0 false negatives.

True. If a model's F1 score is 1, then its accuracy is 100%, which means there cannot be any false positives or false negatives (errors).

H. If you have a word2vec neural network, with **V** total unique words in your entire vocabulary, and are trying to train word embeddings of size **M** dimensions, the output of the **softmax layer** of word2vec is of shape **M x 1**.

*False. The output of the softmax layer of word2vec is V x 1. Remember, the true **y_true** target is a V x 1 vector, with all 0s, except for 1 for the one element that represents the context word. The output of the softmax layer in word2vec is your **y_pred**, your prediction for y that you try to make as similar to **y_true** as possible through backpropagation.*