Q2 - Week 1 questions

• (Q 2.1, Regular Expressions I)
Suppose we want to write a RegEx that only matches consecutive occurrences of 2 or more digits. Which of the following expressions satisfies the purpose? Select all that apply.

(a) `[0-9]*[0-9]+`
(b) `[0-9][0-9]+`
(c) `[0-9][0-9]*`
(d) `[0-9]{2,}`

The answer is (b) (d). The other two options match 1 or more digits.

• (Q 2.2, Regular Expressions II)
Which examples are captured by the RegEx given below?

(I want|Can I have) (some|a lot) ?(of)? ?(those)? (apples |oranges|cherries)([.?])

Select all that apply.

(a) "I want some apples?"
(b) "I don’t want a lot of those cherries."
(c) "Can I have some of those apples?"
(d) "I want a lot of cherries."
(e) "I want some apples and cherries."
(f) "Can I have a lot oranges?"

The answer is (a) (c) (d) (f). The other two options do not match the regex. in the case of (b) "don’t" isn’t a captured word, and in the case of (e) "and" isn’t captured

• (Q 2.3, Tokenization)
When we count the number of words in a corpus, it is important to differentiate between word tokens and word types. In a given corpus, the number of tokens is ______ the number of types.
Which expression below should replace ______?

(a) always greater than or equal to
(b) sometimes greater than and sometimes smaller than
(c) always smaller than or equal to
(d) exactly the same as

The answer is (a). Recall that types are distinct tokens. The number of tokens is what we usually refer to as "word count", and the number of types is what we usually refer to as "vocabulary size".

• (Q 2.4, Edit distance)

Supposing the insertion, deletion, and substitution of a letter all have the SAME cost, how many minimum edit distance alignments are there between words ham and spam?

(In other words, in how different ways can you edit ham into spam?)

(a) 1
(b) 2
(c) 3
(d) 4

The answer is (b).

The modified formula is

\[ D[i, j] = \min \begin{cases} 
D[i-1, j] + 1 \\
D[i, j-1] + 1 \\
D[i-1, j-1] + \begin{cases} 
1; & \text{if } source[i] \neq target[j] \\
0; & \text{if } source[i] = target[j] 
\end{cases}
\end{cases} \]

Thus we get the alignment table:

<table>
<thead>
<tr>
<th></th>
<th>m</th>
<th>a</th>
<th>h</th>
<th>#</th>
<th>s</th>
<th>p</th>
<th>a</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>s</td>
<td>p</td>
<td>a</td>
<td>m</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The two minimum alignments are:

\[
\text{spam spam} \\
| | | | | | | |
* ham h * am
\]

Q3 - Week 2 questions

• (Q 3.1, Language Modeling)

The following table shows the bigram probabilities for eight words in the Berkeley Restaurant Project corpus. For example, using the table we can see that:

\[ P(\text{eat}|\text{to}) = .28 \text{ and } P(\text{to}|\text{eat}) = .0027 \]
Compute the probability of the sentence *i want chinese food* using the appropriate bigram probabilities above. Do not worry about smoothing and remember to add a start and end token to the sentence!

Here are a few other useful probabilities, although you may not need all of them:

\[
\begin{align*}
P(i|<s>) &= .25 \\
P(\text{food}|<s>) &= .68 \\
P(<s>|\text{food}) &= .72 \\
P(<s>|i) &= .02
\end{align*}
\]

Round your answer to 5 digits after the decimal point.

The answer is \textbf{0.00020}.

Since we are already given the bigram probabilities, we just need to model the given sentence as a chain of consecutive bigrams and multiply the corresponding probabilities.

\[
P(i \text{ want chinese food}) = P(i|<s>) \times P(\text{want}|i) \times P(\text{chinese}|\text{want}) \times P(\text{food}|\text{chinese}) \times P(<s>|\text{food})
\]

\[
= .25 \times .33 \times .0065 \times .52 \times .72
\]

\[
= .00020
\]

• (Q 3.2, Text Classification: Naive Burgers I)

Welcome to the best student spot on campus, The Axe and Palm. The Example column in the following corpus contains the condiments that students asked for, after ordering a hamburger or a hotdog, which is shown in the Label column.

<table>
<thead>
<tr>
<th>Example</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can I have lettuce ketchup and mayo</td>
<td>burger</td>
</tr>
<tr>
<td>Can I have ketchup and mustard</td>
<td>hotdog</td>
</tr>
<tr>
<td>Can I have mayo and ketchup</td>
<td>hotdog</td>
</tr>
<tr>
<td>Can I have relish and mustard</td>
<td>burger</td>
</tr>
<tr>
<td>Can I have sriracha onions and jalapenos</td>
<td>hotdog</td>
</tr>
</tbody>
</table>

Table 1: Corpus for Naive Bayes model

Note that the label words (burger and hotdog) do not count as words in the corpus vocabulary. Assume that we have the following stop words: [Can, I, have, and].
Using Naive Bayes with unigram modeling and Laplace smoothing, calculate the probability for someone to ask for ketchup given that they ordered a hotdog. Do not incorporate words in the stop word list in your calculations.

(a) \( \frac{2}{7} \)
(b) \( \frac{1}{5} \)
(c) \( \frac{1}{7} \)
(d) \( \frac{2}{15} \)

The answer is (b).

We first begin by setting up our equation. We are asked to use Naive Bayes with unigram modeling and Laplace smoothing.

\[
P(\text{ketchup}|\text{hotdog}) = \frac{\text{count}(\text{ketchup, hotdog}) + 1}{\Sigma_{w \in V} \text{count}(w, \text{hotdog}) + |V|}
\]

From our table we see that

\[
\text{count}(\text{ketchup, hotdog}) = 2 \\
\Sigma_{w \in V} \text{count}(w, \text{hotdog}) = 7 \\
|V| = 8
\]

Therefore we see that:

\[
P(\text{ketchup}|\text{hotdog}) = \frac{2 + 1}{7 + 8} = \frac{3}{15} = \frac{1}{5}
\]
Assume that we have a large collection of documents, where 10 documents in the collection are relevant for a particular query, while all the other documents are irrelevant. Our goal is to build a model that retrieves all the relevant documents and none of the irrelevant documents. Which of the following set of documents retrieved by our model has a precision of 80% and a recall of 40%? R denotes a relevant document and N denotes an irrelevant document.

Note that we’re asking about precision and recall, not average precision.

(a) N R N R N
(b) R N N R R N N N R N
(c) N R R R R R R R R N
(d) R R N N R

Tip: For example, the sequence "N N R" means that our retriever searched all of the corpus and predicted that there were only three relevant documents. The true labels of two of these documents were N, meaning that our retriever made a prediction error. In addition, our retriever only retrieved one relevant document, missing the other nine.

The answer is (d). Recall that the precision of the model refers to the number of relevant documents retrieved divided by the total number of documents retrieved, while the recall of the model refers to the number of relevant documents retrieved divided by the total number of relevant documents in the collection (which is 10 in this case).

Option (a) is incorrect because it has a precision of \( \frac{2}{5} = 40\% \) and a recall of \( \frac{2}{10} = 20\% \).

Option (b) is incorrect because it has a precision of \( \frac{4}{10} = 40\% \) and a recall of \( \frac{4}{10} = 40\% \).

Option (c) is incorrect because it has a precision of \( \frac{8}{10} = 80\% \) and a recall of \( \frac{8}{10} = 80\% \).

Option (d) is correct because it has a precision of \( \frac{4}{5} = 80\% \) and a recall of \( \frac{4}{10} = 40\% \).
Q4 - Week 3 questions

• (Q 4.1, Query Optimization)
You are given a set of 450 documents and the following document frequencies for some terms (see Table 2):

<table>
<thead>
<tr>
<th>Term</th>
<th>Document Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>bike</td>
<td>350</td>
</tr>
<tr>
<td>tree</td>
<td>400</td>
</tr>
<tr>
<td>COHO</td>
<td>150</td>
</tr>
<tr>
<td>MemChu</td>
<td>200</td>
</tr>
<tr>
<td>Tresidder</td>
<td>100</td>
</tr>
<tr>
<td>Cardinal</td>
<td>250</td>
</tr>
</tbody>
</table>

Table 2: Document frequencies for query optimization

Given the following query:

bike AND tree AND (MemChu OR NOT Tresidder) AND NOT (COHO OR Cardinal)

Which of the following parts should we execute first?

(a) bike AND tree
(b) MemChu OR NOT Tresidder
(c) NOT (COHO OR Cardinal)
(d) NOT Tresidder AND NOT COHO

The answer is (c). Remember that if a term x occurs in X documents and a term y occurs in Y documents, x AND y will return at most max X, Y documents, x OR y will return at most X + Y documents, and NOT x will return at most N - X documents (where N is the total number of documents). Because the query in the question is a number of subqueries chained together with ANDs, we want to execute the part of the query that will return the smallest number of documents first.

bike AND tree → max {350, 400} = 350.

MemChu OR NOT Tresidder → 200 + (450 - 100) = 550. (Note that there are only 450 documents, so this this will return at most 450 documents).

NOT (COHO OR Cardinal) → 450 - (max {150, 250}) = 200. (Note that the maximum number of documents for COHO OR Cardinal would be 150 + 250 = 400, but because we want the maximum number of documents after negation, we need to consider the minimum number of documents for COHO OR Cardinal, which is 250.)

NOT Tresidder AND NOT COHO. This is not a part of the query (it ignores the parentheses), so it should not be executed.

So, 200 is the lowest number of documents, so the answer is (c).

• (Q 4.2, Inverted Positional Index I)
Use the following inverted positional index for the next three questions.
"be" ==> { 1: [3], 2: [1], 3: [2], 4: [2, 4] }
"do" ==> { 1: [5], 2: [2, 4], 3: [5], 4: [1, 3, 5] }
"is" ==> { 1: [1], 2: [3], 3: [3] }
"to" ==> { 1: [2, 4], 2: [5], 3: [1, 4] }

Note: for the purposes of this question, documents are 1-indexed (the first word in the document is at index 1).

Which of the following documents was NOT used to create the inverted positional index shown above?

(a) Doc 1: “is to be to do”
(b) Doc 2: “to do is to be”
(c) Doc 3: “to be is to do”
(d) Doc 4: “do be do be do”

The answer is (b). If a document was used to create the positional index, then for each word in that document, the entry for that word in the index should contain the document number and the positions of the word in the document (i.e. word ==> {doc_number:[position1, position2,...]}).

Doc 1 has the word "is" at index 1, the word "to" at indices 2 and 4, the word "be" at index 3, and the word "do" at index 5. This matches the position values for document 1 in each of the words’ entries in the index, which means Doc 1 was in fact used in creating the index.

Doc 3 has the word "to" at indices 1 and 4, the word "be" at index 2, the word "is" at index 3, and the word "do" at index 5. This matches the position values for document 3 in each of the words’ entries in the index, which means Doc 3 was in fact used in creating the index.

Doc 4 has the word "do" at indices 1, 3, and 5, and the word "be" at indices 2 and 4. This matches the position values for document 4 in each of the words’ entries in the index, which means Doc 4 was in fact used in creating the index.

Doc 2 has the word "to" at indices 1 and 4, the word "do" at index 2, the word "is" at index 3, and the word "be" at index 5. This does not match the position values for document 2 in all of the words’ entries in the index: although the index reports that the word "do" appears in Doc 2 at index 2 and that the word "is" appears at index 3, it also reports that the words "be", "do", and "to" appear in Doc 2 at indices 1, 4, and 5, respectively, which is not true. Therefore we know that Doc 2 was not used in creating the inverted index.

• (Q 4.3, Inverted Positional Index II)

Based on the inverted positional index shown above, what is the content of Doc 2?

(a) Doc 2: “be do is do to”
(b) Doc 2: “to do be be”
(c) Doc 2: “is to do to be”
(d) None of the above

The answer is (a). The inverted positional index reports that Doc 2 contains the word "be" at index 1, the word "do" at indices 2 and 4, the word "is" at index 3, and the word "to" at index 5. This means that Doc 2 is "be do is do to".
Select all of the **phrase queries** that would return at least one document:

(a) "to is"
(b) "do is"
(c) "be is"
(d) "be do"

The answer is (b) (c) and (d). We know that the four documents in our collection are:

- Doc 1: “is to be to do”
- Doc 2: “be do is do to” (from part 4.3)
- Doc 3: “to be is to do”
- Doc 4: “do be do be do”

Option (b) is correct since the phrase "do is" can be found in Doc 2. This can also be shown using the inverted positional index: at least one document (Doc 2) contains the words "do" and "is" in that order at consecutive indices.

Option (c) is correct since the phrase "be is" can be found in Doc 3. This can also be shown using the inverted positional index: at least one document (Doc 3) contains the words "do" and "is" in that order at consecutive indices.

Option (d) is correct since the phrase "be do" can be found in Doc 2 and Doc 4. This can also be shown using the inverted positional index: at least one document (Doc 2 and Doc 4) contains the words "do" and "is" in that order at consecutive indices.
### Q5 - Week 4 questions

- (Q 5.1, TF-IDF)

Given the following term frequencies (counts) for a few words in a collection of 4 documents,

<table>
<thead>
<tr>
<th>Term</th>
<th>Doc 1</th>
<th>Doc 2</th>
<th>Doc 3</th>
<th>Doc 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>seashell</td>
<td>34</td>
<td>30</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>ocean</td>
<td>39</td>
<td>12</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>surf</td>
<td>26</td>
<td>0</td>
<td>13</td>
<td>18</td>
</tr>
<tr>
<td>sunshine</td>
<td>0</td>
<td>5</td>
<td>17</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3: Term counts per document.

Use the following equations to compute **tf-idf**, and assume that these are the only documents and words in the collection.

\[
\text{tf}_{t,d} = \log_{10}(\text{count}(t, d) + 1)
\]

\[
\text{idf}_t = \log_{10}(N/df_t)
\]

\[
\text{tf-idf}_{t,d} = \text{tf}_{t,d} \cdot \text{idf}_t
\]

Where \(\text{count}(t, d)\) is the number of occurrences of term \(t\) in \(d\), \(N\) is the total number of documents, and \(df_t\) is the number of documents containing \(t\).

Which one of the above words have the **highest** tf-idf weighting in **Doc 2**?

(a) seashell  
(b) ocean  
(c) surf  
(d) sunshine

The answer is **(d)**. Let’s begin by calculating all of the \(\text{tf-idf}_{t,d}\) weightings for Doc 2 and each of the four terms: seashell, ocean, surf, and sunshine. Using the equations provided, we find that,

- \(\text{tf-idf}_{\text{seashell},2} = \log_{10}(30 + 1) \cdot \log_{10}(\frac{4}{3}) = .186\)
- \(\text{tf-idf}_{\text{ocean},2} = \log_{10}(12 + 1) \cdot \log_{10}(\frac{4}{3}) = .139\)
- \(\text{tf-idf}_{\text{surf},2} = \log_{10}(0 + 1) \cdot \log_{10}(\frac{4}{3}) = 0\)
- \(\text{tf-idf}_{\text{sunshine},2} = \log_{10}(5 + 1) \cdot \log_{10}(\frac{4}{3}) = .234\)

Therefore, since .234 is the greatest tf-idf weighting in Doc 2, we know that **(d) sunshine** is the correct answer.