

## Classical text mining Point Quiz, 5 questions

5/5 points (100%)

Imagine you have a texts database. Here are stemming and lemmatization results for some of the words:

Word	Stem	Lemma
operate	oper	operate
operating	oper	operating
operates	oper	operates
operation	oper	operation
operative	oper	operative
operatives	oper	operative
operational	oper	operational

Imagine you want to find results in your texts database using the following queries:

1. operating system (we are looking for articles about OS like Windows or Linux)

2. **operates in winter** (we are looking for machines that can be operated in winter) Before execution of our search we apply either stemming or lemmatization to both query and texts. Compare stemming and lemmatization for a given query and choose the correct statements. Stemming provides higher F1-score for operating system query. **Un-selected is correct** Stemming provides higher precision for operating system query. **Un-selected is correct** Lemmatization provides higher precision for operates in winter query. Correct This is true, but it would loose a lot of other relevant forms. Stemming provides higher recall for operates in winter query.

## Correct

This is true, lemmatization would only find exact matches with operates and lose a lot of relevant forms like operational.

20/04/2019, 18:08 2 di 5

Classical text mining Quiz, 5 questions	5/5 points (100%
3. Choose correct statements about bag-of-words (or n-grams) features.	
Classical bag-of-words ${f vectorizer}$ (object that does vectorization) needs an amount of RAM proportional to $T$ , which is the number of unique tokens in the dataset.	at least
<b>Correct</b> This is true, you have to store a hash map {token: index} to be able to vectorize new texts.	
You get the same vectorization result for any words permutation in your text.	
Un-selected is correct	
Hashing <b>vectorizer</b> (object that does vectorization) needs an amount of RAM proportional t to operate.	o vocabulary size
Un-selected is correct	
For bag-of-words features you need an amount of RAM at least proportional to $N  imes T$ , who number of documents, $T$ is the number of unique tokens in the dataset.	ere $N$ is the
Un-selected is correct	
We prefer <b>sparse</b> storage formats for bag-of-words features.	
<b>Correct</b> This is true. We have a lot of zeros in these features, that's why we can store them efficiently in sp (look at sklearn.feature_extraction.text.TfidfVectorizer and scipy.sparse.csr.csr_matrix).	oarse formats
✓ 1/1	

4.

Let's consider the following texts:
Classical text mining
Quigotopherions

5/5 points (100%)

- not a good movie
- did not like
- i like it
- good one

Let's count **Term Frequency** here as a distribution over tokens in a particular text, for example for text "good one" we have TF = 0.5 for "good" and "one" tokens.

## Term frequency (TF)

- tf(t, d) frequency for term (or n-gram) t in document d
- · Variants:

weighting scheme	TF weight
binary	0,1
raw count	$f_{t,d}$
term frequency	$f_{t,d}/\sum_{t'\in d}f_{t',d}$
log normalization	$1 + \log(f_{t,d})$

## Inverse document frequency (IDF)

- N = |D| total number of documents in corpus
- $|\{d \in D: t \in d\}|$  number of documents where the term t appears
- $idf(t,D) = log \frac{N}{|\{d \in D: t \in d\}|}$

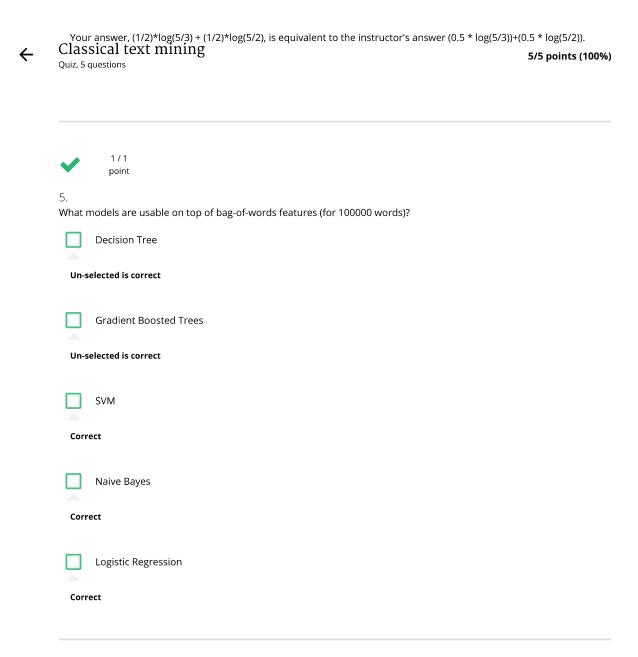
What is the **sum** of TF-IDF values for 1-grams in "good movie" text? Enter a math expression as an answer. Here's an example of a valid expression: log(1/2)\*0.1.

Preview

$$-rac{1}{2}\log\left(3
ight)-rac{1}{2}\log\left(2
ight)+\log\left(5
ight)$$

(1/2)\*log(5/3) + (1/2)\*log(5/2)

**Correct Response** 



3 P