←

Classical text mining

Quiz, 5 questions

1 point

1. Choose true statements about text tokens.

- [ ] Stemming can be done with heuristic rules
- [ ] Lemmatization needs more storage than stemming to work
- [ ] Lemmatization is always better than stemming
- [ ] A model without stemming/lemmatization can be the best

---

1 point

2. 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.

- [ ] Lemmatization provides higher precision for **operates in winter** query.
- [ ] Stemming provides higher F1-score for **operating system** query.
- [ ] Stemming provides higher precision for **operating system** query.
- [ ] Stemming provides higher recall for **operates in winter** query.

---

1 point

3. Choose correct statements about bag-of-words (or n-grams) features.

- [ ] Classical bag-of-words **vectorizer** (object that does vectorization) needs an amount of RAM at least proportional to T, which is the number of unique tokens in the dataset.
- [ ] You get the same vectorization result for any words permutation in your text.
- [ ] For bag-of-words features you need an amount of RAM at least proportional to $N \times T$, where N is the number of documents, T is the number of unique tokens in the dataset.
- [ ] We prefer **sparse** storage formats for bag-of-words features.
- [ ] Hashing **vectorizer** (object that does vectorization) needs an amount of RAM proportional to vocabulary size to operate.

---

1 point

4. Let's consider the following texts:

- good movie
- 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)

- $\mathrm{tf}(t, d)$ – 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
- $\text{idf}(t, D) = \log \frac{N}{|\{d \in D: t \in d\}|}$

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

$$-0.5 \log(6) + 0.5 \log(25)$$

0.5*log(25/6)

1 point

5. What models are usable on top of bag-of-words features (for 100000 words)?

- [ ] Logistic Regression
- [ ] Naive Bayes
- [ ] Gradient Boosted Trees
- [ ] Decision Tree
- [ ] SVM

- [x] I, **HYUNSUP YOON**, understand that submitting work that isn't my own may result in permanent failure of this course or deactivation of my Coursera account.

Learn more about Coursera's Honor Code

Submit Quiz