Spark MLlib

Training Session

Text Analytics – TF-IDF

- Term frequency-inverse document frequency (TF-IDF) is a method widely used in text mining to reflect the importance of a term to a document in the corpus.
- It is a way of feature vectorization i.e. representing a document as a vector of the terms present in the document.
- Denote a term by t, a document by d, and the corpus by D.
- Term frequency TF(t,d) is the number of times that term t appears in a document d.
- Document frequency DF(t,D) is the number of documents that contains term t.
- If we only use term frequency to measure the importance, it is very easy to over-emphasize terms that appear very often but carry little information about the document, e.g. “a”, “the”, and “of”.
- If a term appears very often across the corpus, it means it doesn’t carry special information about a particular document.
- Inverse document frequency is a numerical measure of how much information a term provides: $IDF(t,D) = \log \frac{|D|+1}{DF(t,D)+1}$

References:

https://spark.apache.org/docs/2.4.0/ml-features.html
https://spark.apache.org/docs/2.4.0/api/python/pyspark.ml.html#pyspark.ml.feature.IDF

Text Analytics – TF-IDF

- Since logarithm is used, if a term appears in all documents, its IDF value becomes 0.
- Note that a smoothing term is applied to avoid dividing by zero for terms outside the corpus.
- The TF-IDF measure is simply the product of TF and IDF: $TFIDF(t, d, D) = TF(t, d) \cdot IDF(t, D)$.
- There are several variants on the definition of term frequency and document frequency.
- In Spark MLlib, TF and IDF are separated to make them flexible.
- HashingTF and CountVectorizer can be used to generate the term frequency vectors.
- CountVectorizer and CountVectorizerModel aim to help convert a collection of text documents to vectors of term counts.
  - When an a-priori dictionary is not available, CountVectorizer can be used as an Estimator to extract the vocabulary, and generates a CountVectorizerModel.
  - The model produces sparse representations for the documents over the vocabulary, which can then be passed to other algorithms.
- HashingTF converts documents to vectors of fixed size.
  - The default feature dimension is 262,144 (2^{18}).
  - The terms are mapped to indices using a Hash Function.

References:

https://spark.apache.org/docs/2.4.0/api/python/pyspark.ml.html#pyspark.ml.feature.CountVectorizer
https://spark.apache.org/docs/2.4.0/api/python/pyspark.ml.html#pyspark.ml.feature.CountVectorizerModel
https://spark.apache.org/docs/2.4.0/api/python/pyspark.ml.html#pyspark.ml.feature.HashingTF
https://towardsdatascience.com/countvectorizer-hashingtf-e66f169e2d4e?gi=639998791b66

Text Analytics – TF-IDF

- Considering a simple example

```
+---+--------------------+
|id |words               |
+---+--------------------+
|0  |[PYTHON, HIVE, HIVE]|  → Document 1
|1  |[JAVA, JAVA, SQL]   |  → Document 2
+---+--------------------+
                            → Corpus
```

IDF(PYTHON,Document 1) = log((2+1)/(1+1)) ~ 0.405
IDF(HIVE,Document 1) ~0.405

TF-IDF(PYTHON,Document 1,Corpus) ~ 0.405
TF-IDF(HIVE,Document 1, Corpus) ~0.81

- CountVectorizer extracts vocabulary. And generates a vector for each document.

Term	Frequency	Index
HIVE	2	0
JAVA	2	1
SQL	1	2
PYTHON	1	3

```
+---+--------------------+-------------------------+
|id |words               |features                 |
+---+--------------------+-------------------------+
|0  |[PYTHON, HIVE, HIVE]|(4,[0,3],[2.0,1.0])      |
|1  |[JAVA, JAVA, SQL]   |(4,[1,2],[2.0,1.0])      |
+---+--------------------+-------------------------+
```

Vector Length
Vector Indices
Vector Values

- HashingTF generates a vector for each document as below

```
+---+--------------------+----------------------------------------+
|id |words               |features                                |
+---+--------------------+----------------------------------------+
|0  |[PYTHON, HIVE, HIVE]|(262144,[129668,134160],[2.0,1.0])|
|1  |[JAVA, JAVA, SQL]   |(262144,[53343,167238],[2.0,1.0]) |
+---+--------------------+----------------------------------------+
```