Challenge 3: Inverted index for Wikipedia

Your task is to implement a Spark application that computes an inverted index [1] based on a dump of Wikipedia articles [2].

In particular, you need to map each word onto the list of articles where it occurs, along with the number of occurrences. The final result should have the following format:

(word 1, ((article title 1, occurrences in article 1), (article 2, occurrences in 2)...))
(word 2, ((article 3, occurrences in 3)...))
...

The dataset

We will be using a fraction of the Wikipedia dump as the dataset.

- First, you need to download a chunk of your choice from [2]. (E.g. https://dumps.wikimedia.org/enwiki/20161120/enwiki-20161120-pages-meta-current1.xml-p000000010p000030303.bz2)
-
- Second, convert the XML to a usable format. Use WikiExtractor.py, found here [3].
- It is sufficient if you use a fraction of the generated output.

The data set contains full articles in plain text, with the articles' name prepended to each line.

Implementation hints

- Ignore non-alphanumeric characters. You can use the following pattern to split each line into words:

  Pattern NON_ALPHANUMERIC = Pattern.compile("[^a-zA-Z0-9']");
- Convert all words to lowercase.
-
- In the late stages of your Spark job, you should have an RDD of the type JavaPairRDD<String, Tuple2<String, Integer>>, which represents (word, (article, occurrences)) tuples. The final step toward generating the required result can be achieved by grouping the tuples by key:

  JavaPairRDD<String, Tuple2<String, Integer>> preFinalRDD = ...; //(word, (article, occurrences))

  JavaPairRDD<String, Iterable<Tuple2<String, Integer>>> finalRDD = preFinalRDD.groupByKey(); //(word, ((article 1, occurrences 1), (article 2, occurrences 2)...))

[1] https://en.wikipedia.org/wiki/Inverted_index
[2] https://dumps.wikimedia.org/enwiki/20161120/ (enwiki-20161120-pages-meta-current*.xml-p*.bz2)

[3] https://github.com/vlolteanu/wikiextractor/tree/atds (hint: use "-b atds" when cloning)