Xiao's Solution

Tools and Libraries used

Java 1.7, Apache Maven, Apache Commons Codec, Apache Commons lang, jUnit 4.11;

To run the program

Run the test classes in src/test/java/xiaozhou3;

There are 3 test classes test 3 different modules in main file, and a MainTest to test the whole program. Cancel the annotation in test classes will print the result;

- CandidatesProducerTest.java test CandidatesProducer.java, this class is supposed to read from a dictionary and produce a potential candidate queue;
- CandidatesConsumerTest.java test CandidatesConsumer.java, this class is supposed to poll the candidates queue, filter each candidate using PunFilter.java and produce a pun list;
- PunFilterTest.java test PunFilter.java, this class has two main functions, findPun() and storePun(). findPun() is supposed to determine whether the candidate is a pun, and avoid sending duplicate word (e.g. POPular(popular) and POPlar(popular)) to queue; storePun() is supposed to store puns into a priority queue as well as eliminate redundant puns (e.g. DOGs(dogs) and DOGs(docs)). Both two methods can be tested in PunFilterTest.java;
- MainTest.java test the whole program. test() and test2() are included in testing.

```
@Test(timeout = 1000)
public void test() {
    String target = "DOG";
    String fileName = "DICTIONARY";
    int maxPuns = 1000;
    CandidatesConsumer cc = new CandidatesConsumer(target, maxPuns);

    cc.parseQueue(CandidatesProducer.readFrom(target, fileName));

    assertNotNull(cc.getResult());
    assertNotEquals(cc.getResult().size(), 0);

    for(String pun:cc.getResult())
        assertTrue(pun.contains(target));

    print(cc.getResult(), target);
}
```

Figure1, MainTest.test() is used to test the program with user-defined word. '$target' could be a word even not appeared in the dictionary. '$fileName' is the name of your dictionary and maxPuns is the number of puns being showed;

```java
@Test
public void test2() {
    String fileName = "DICTIONARY";
    List<String> list = readList(fileName);
    int maxPuns = 10;

    for(String target:list) {
        target = target.toUpperCase();
        CandidatesConsumer cc = new CandidatesConsumer(target, maxPuns);

        cc.parseQueue(CandidatesProducer.readFrom(target, fileName));

        assertNotNull(cc.getResult());

        for(String pun:cc.getResult())
            assertTrue(pun.contains(target));

        print(cc.getResult(), target);
    }
}
```

Figure2, MainTest.test2() is used to test the program with several iterations. When got a list of target words, test2() test all of the words and judge whether error happened;

Steps

1. CandidatesProducer.java scans the dictionary, puts potential candidates in a queue;
2. CandidatesConsumer.java consumes the queue, sends candidates to PunFilter.java to filters candidates. Qualified puns will be store in a sorted queue;
3. CandidatesConsumer.java then converts the priority queue to a result list;

Function of this approach

1. Sorted puns list using priority queue;
2. Redundant puns are removed; (e.g. DOGs(dogs) and DOGs(docs) have the same pun name while only DOGs(dogs) is remained because it is closer to DOG);
3. Redundant words are removed; (POPularity(popularity) and POPlarity(popularity) are both puns of POP, only POPularity(popularity) will be selected though because it is closer to POP);
4. Metaphone phonetic encoding scheme and Levenshtein distance calculation help improve the accuracy;
5. Auto test program;