Software Package: Ripley Speechtools Analytics
User Manual version 0.0.1

1. Introduction

a) Motivation – The motivation for the Ripley Speechtools Analytics library is a need by the speech processing and recognition community to perform tasks in a distributed environment to manage the increasing amount of available data for training and testing purpose.
b) Goals – The initial goal for the RSA library is to utilize existing, open-source libraries to perform distributed training for an n-gram language model targeted for speech recognition applications. This is part of the overall project goal to develop capabilities required to condition, decode, and understand large volumes of speech content.
c) History and Time Line – The project was started in October 2014, and first uploaded to a public repository in December 2014 with a first implementation of distributed language model training with preexisting open source software packages.
d) People – The project is maintained by Kyle White (kjw2146@columbia.edu).
e) Licensing – The open-source toolkits utilized by the RSA project include {berkelylm, Apache Lucene, Apache Hadoop}. All these open source libraries utilize the Apache License, version 2.0, though Hadoop and Lucene have been directly submitted to ASF and berkeleylm is otherwise maintained. An Apache License, version 2.0, also applies to the RSA library. The LICENE and NOTICE documents are in the top-level directory of the project, per ASF guidelines.

2. Implementation

a) Overview – The project currently consists of only one application to train a n-gram language model, and the implementation is currently spread across two phases. In the first phase, Hadoop is utilized to count n-grams across the input transcriptions in a distributed manner. The intermediate output is a n-gram count document in a simple, RSA internal format.
b) Phase I – The phase I implementation is centered around usage of the Hadoop distributed computing and Lucene text analysis libraries available as top level, open-source Apache projects. The Hadoop library is utilized to created a Java MapReduce application capable of taking textual transcriptions as input and producing output n-gram count documents. Custom Client, Mapper, Reducer, and Partitioner classes are utilized in a Maven project to enable usage of the MapReduce paradigm via the Hadoop libraries. The embedded Javadoc contains information regarding the data flow and further setup of the Hadoop related application details. Lucene core library capabilities for analysis/tokenization are utilized by the custom Mapper classes to perform analysis on the input for n-gram creation from transcript strings. The data pipeline of Lucene TokenStream objects utilized for tokenization is outlined in the Javadoc for related RSA classes. Usage of Lucene allows the use of standard Unicode segmentation and further filtering on the input transcripts.
c) Phase II – The phase II implementation is centered around usage of the berkeleylm library for creation of large language models and efficient language model access.

Usage of this library aligns with the goals of the RSA project due to a common focus on large data sets. Beyond the ability to efficiently scale up to large data sets, the realism behind speech recognition and synthesis systems is partly determined by realistic system reaction times. For example, if the latency between such a system receiving a user utterance and replying with a result is to large, this may decrease system acceptance. The decoding time of speech recognition systems is commonly larger for systems utilizing statistical language models (SLM) compared to command grammars, and can be even more so with SLMs that span a large vocabulary and/or domain, and attempt to search a significant sub-graph of this space due to tough requirements on decoding accuracy. The ability of berkeleylm to efficiently store and access language modeling information represents a possible boon to mitigating this latency concern.

3. Usage

a) Overview – Usage of the library is currently a two phase affair, where all operations are performed on files in HDFS. Simplification to a single phase is planned. The library is built via Maven, and there is currently one additional step required beyond calling 'maven clean install' after downloading the repository. The Maven pom depends on a berkelylm v1.1.5 JAR that is not currently in the Maven central repository. It must be downloaded and locally installed via Maven using the parameters as specified in the pom.xml file downloaded with RSA.

b) Phase I – As the implementation section discussed, phase I will take the input transcripts and output N n-gram count documents, one for each order of the n-gram being trained. An example script is show below to utilize the command line entry point for phase I processing.

```
#!/bin/bash

# This script is written to execute the speechtools library class LMTrainer
# The LMTrain is inteded to initiate a Hadoop MapReduce job to perform
# n-gram language model training on data in HDFS
# ${1} - input data path (file or directory) relative to the HDFS
# ${2} - output data path (file or directory) relative to the HDFS for ngram count int
ermediate document

export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:$(hadoop classpath)

export LIBJARS1=~/Software/lucene-4.10.2/analysis/common/lucene-analyzers-common-4.10.
2.jar,~/Software/lucene-4.10.2/core/lucene-core-4.10.2.jar

# Run hadoop job to create intermediate ngram document file via mapreduce job
hadoop --config $HADOOP_CONF_DIR jar ~/workspace/speechtools/target/speechtools-0.0.1-
SNAPSHOT.jar ripley.speechtools.client.LMTrainer -libjars $LIBJARS1 ${1} ${2}
```

c) Phase II - As the implementation section discussed, phase II will take an intermediate n-gram count document and generate an output language model representation in an ARPA format language model. An example script is show below to utilize the command line entry point for phase II processing. Additional details for this steps

are in the associated Javadoc (e.g., details such as how the resulting n-gram is created using Kneser-Ney probability smoothing as implemented in berkeleylm which largely mimics in Java the SRI toolkit implementation).

```
#!/bin/bash

# This script is written to execute the speechtools library class LMTrainer
# The LMTrain is inteded to initiate a Hadoop MapReduce job to perform
# n-gram language model training on data in HDFS
# ${1} - input data path (currently only a file) of intermediate, ordered ngram
count file relative to the HDFS
# ${2} - output data path (file) relative to the HDFS for ARPA format ngram lang
uage model document

export LIBJARS2=~/software/berkeleylm-1.1.5/jar/berkeleylm.jar

hadoop --config $HADOOP_CONF_DIR jar ~/workspace/speechtools/target/speechtools-
0.0.1-SNAPSHOT-jar-with-dependencies.jar ripley.speechtools.LMCompiler.KneserNey
LMCompiler ${1} ${2}
```