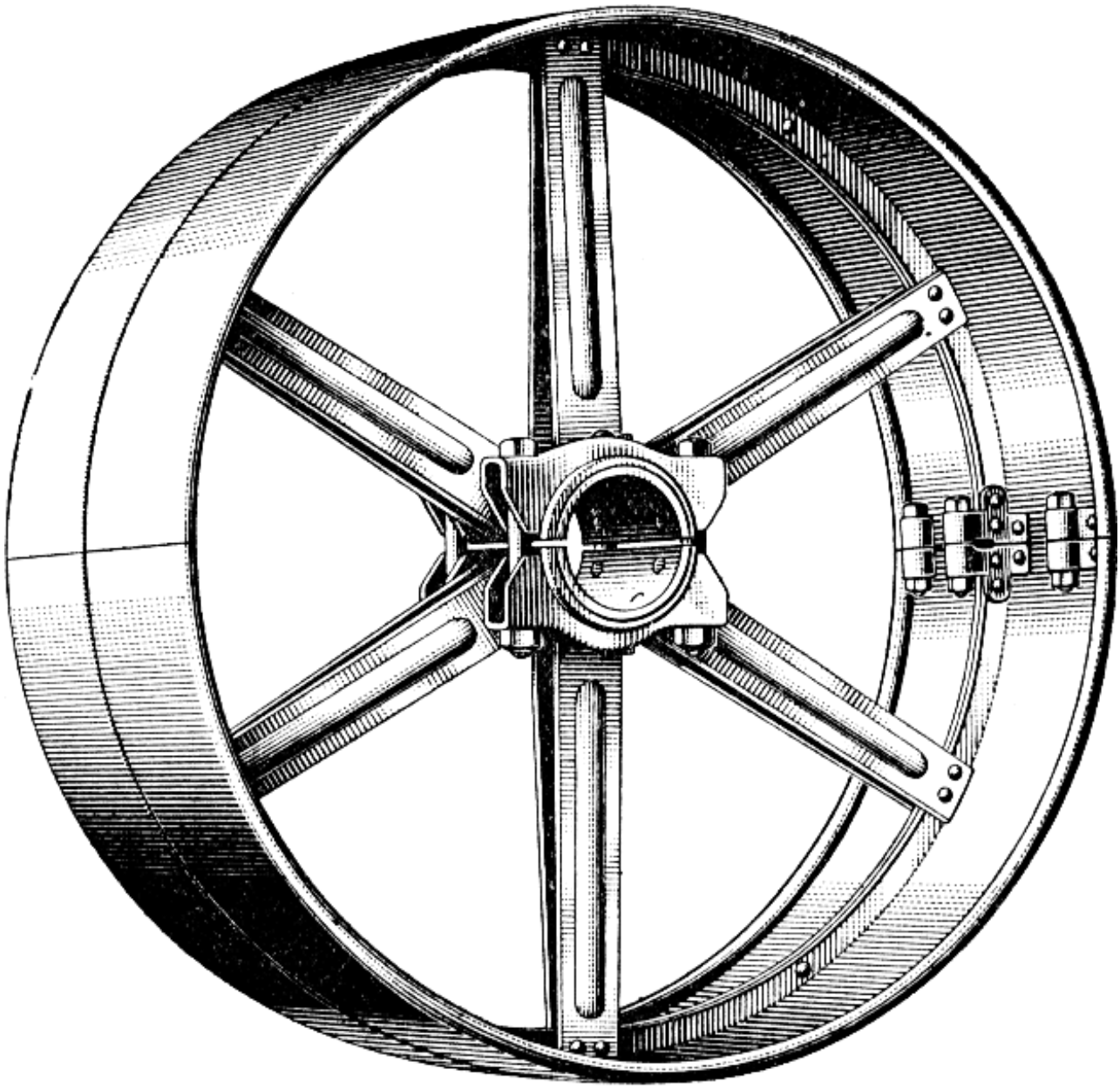

Fig. 16

Rejecto 2.03 Manual

Written by Halle Winkler, published by Politepix

Friday, January 9, 2015

Introduction and Installation

Introduction

Rejecto is an out-of-vocabulary rejection plugin for the LanguageModelGenerator of OpenEars for use with English and Spanish language models.

It is sometimes the case, especially with small vocabularies, that Pocketsphinx will report unrelated words as being matches for its vocabulary. Rejecto adds an out-of-vocabulary rejection model that increases the probability that words which are not in your vocabulary will not be recognized as words which are in your vocabulary. It does this by listening for isolated speech sounds (phonemes, like "AH" and "K") which do not map to the words you are actively listening for, and then rejecting recognitions in which they occur.

Rejecto is compatible with the English acoustic model AcousticModelEnglish.bundle and the Spanish acoustic model AcousticModelSpanish.bundle.

Rejecto can be purchased at the Politepix shophere and we recommend thoroughly evaluating the demo version before purchasing, which can be downloaded from this shop page. The installation and usage process is the same for both the demo and licensed version, but the demo times out after 3 minutes of use and can't be submitted to the App Store. The best way to get started using Rejecto is to get a tutorial from the Politepix interactive tutorial tool. Steps for getting started and more in-depth documentation are also provided on this page.

Installation

How to install and use Rejecto:

Rejecto is a plugin for OpenEars, so it is added to an already-working OpenEars project in order to enable new OpenEars features. In these instructions we are using the OpenEars sample app as an example for adding the plugin and new features, but the steps are basically the same for any app that already has a working OpenEars installation. Please note that

Rejecto requires OpenEars 2.0 or greater. Rejecto also works with RapidEars versions 2.0 or greater.

1. Download and try out the OpenEars distribution and try the OpenEars sample app out Rejecto is a plug-in for OpenEars that is added to an OpenEars project so you first need a known-working OpenEars app to work with. The OpenEars sample app is fine for this to get started. You can also get a complete tutorial on both creating an OpenEars app and adding Rejecto to it using the automatic customized tutorial.
2. Open up the OpenEars Sample App in Xcode. Drag your downloaded RejectoDemo.framework into the OpenEars sample app project file navigator.
3. Open up the Build Settings tab of your app or OpenEarsSampleApp and find the entry "Other Linker Flags" and add the linker flag "-ObjC". Do this for debug and release builds. More explanation of this step can be seen in the tutorial by selecting the "speech recognition which ignores spoken words which aren't in the vocabulary" tutorial, which will also show exactly how to use the new methods added by Rejecto.

Support

You can have one free email support incident with the demo version of Rejecto, and as many questions on the OpenEars plugins forums as you like. To use your free email support incident, you must register your app and a verifiable name (company name or personal name) here.

You can also send as many sales inquiries as you like through the contact form, and you don't need to register in order to do so, although a sales inquiry with a tech support question will be considered a support incident and we'll ask you to register in order to have it engaged.

Once you have completed licensing of the framework for your app, you get more email support incidents if you have purchased an email support bundle, and continued forum support. Extra email support incidents for demo and licensed versions can always be purchased at the Politepix shop. Support contracts for multiple email support incidents with Politepix can also be purchased. Licensing the framework requires giving the exact application name that the framework will be linked to, so don't purchase the license until you know the app name. Please read on for the Rejecto documentation.

OELanguageModelGenerator+Rejecto Category Reference

Detailed Description

A plugin which adds the ability to reject out-of-vocabulary words and statements when using OpenEars or RapidEars speech recognition.

Usage examples

> *What to add to your OpenEars implementation:*

First, find the line

```
#import <OpenEars/OELanguageModelGenerator.h>
```

in your app and add the following line right underneath it:

```
#import <RejectoDemo/OELanguageModelGenerator+Rejecto.h>
```

Next, change this line where you create a language model:

```
NSError *err = [lmGenerator generateLanguageModelFromArray:words
withFilesNamed:name forAcousticModelAtPath:[OEAcousticModel
pathToModel:@"AcousticModelEnglish"]];
```

to use this method instead:

```
NSError *err = [lmGenerator generateRejectingLanguageModelFromArray:words
 withFilesNamed:name
 withOptionalExclusions:nil
 usingVowelsOnly:FALSE
 withWeight:nil
 forAcousticModelAtPath:[OEAcousticModel
pathToModel:@"AcousticModelEnglish"]]; // Change "AcousticModelEnglish" to
"AcousticModelSpanish" to create a Spanish Rejecto model.
```

You will use the same array for languageModelArray (again, the words and phrases in your array must be in all-capital letters) and the same files name for fileName as you did with the old generateLanguageModelFromArray method, and to get started you can use the value

"nil" for optionalExclusions, vowelsOnly, and weight, since they are there to help you refine your results and might not be needed. You can learn more about fine-tuning your results with those optional parameters in the Rejecto documentation.

Warning

> It isn't necessary to use optionalExclusions in order to try to tell Rejecto not to add a phoneme that is equivalent to a word in your vocabulary (for instance, Rejecto will add a phoneme by default that represents the "I" sound in the word "I" and the word "EYE", but if those words (or another word using the "I" sound by itself) are in your vocabulary, it will automatically not add the rejection phoneme that has the "I" sound.

It is only necessary to use optionalExclusions in the uncommon event that there is a phoneme that is being perceived more frequently than it is spoken, to the detriment of your speech detection of words that are really in your vocabulary.

Method Documentation

```
- (NSError *) generateRejectingLanguageModelFromArray: (NSArray *)  languageModelArray
                                       withFilesNamed: (NSString *) fileName
                                 withOptionalExclusions: (NSArray *)  optionalExclusions
                                       usingVowelsOnly: (BOOL)        vowelsOnly
                                            withWeight: (NSNumber *) weight
                                 forAcousticModelAtPath: (NSString *) acousticModelPath
```

This is the method which replaces OpenEars' OELanguageModelGenerator's generateLanguageModelFromArray: method in a project which you have added this plugin to. It generates a language model from an array of NSStrings which are the words and phrases you want OEPocketsphinxController or OEPocketsphinxController+RapidEars to understand. Putting a phrase in as a string makes it somewhat more probable that the phrase will be recognized as a phrase when spoken.

fileName is the way you want the output files to be named, for instance if you enter "MyDynamicLanguageModel" you will receive files output to your Caches directory titled MyDynamicLanguageModel.dic and MyDynamicLanguageModel.DMP.

If this method is successful it will return nil. If it returns nil, you can use the methods pathToSuccessfullyGeneratedDictionaryWithRequestedName: and pathToSuccessfullyGeneratedLanguageModelWithRequestedName: or pathToSuccessfullyGeneratedGrammarWithRequestedName: to get your paths to your newly-generated language models and grammars and dictionaries for use with OEPocketsphinxController. If it doesn't return nil, it will return an error which you can check for debugging purposes.

The words and phrases in languageModelArray must be written with capital letters exclusively, for instance "word" must appear as "WORD". You pass in the path to the acoustic model you want to use, e.g. [OEAcousticModel

pathToModel:"AcousticModelEnglish"] or [OEAcousticModel pathToModel:@"AcousticModelSpanish"] which are currently the only two acoustic models which work with Rejecto.

optionalExclusions can either be set to nil, or given an NSArray of NSStrings which contain phonemes which you do not want to have added to the rejection model. A case in which you might want to do this is when you have over-active rejection such that words that are really in the vocabulary are being rejected.

You can first turn on deliverRejectedSpeechInHypotheses: in order to see which phonemes are being detected overzealously and then you can add them to the exclusionArray. Set this parameter to nil if you aren't using it.

usingVowelsOnly allows you to limit the rejection model to only vowel phonemes, which should improve performance in cases where that is desired. Set this parameter to FALSE if you aren't using it.

The last optional parameter is weight, which should usually be set to nil, but can also be set to an NSNumber with a floatValue that is greater than zero and equal or less than 2.0. This will increase or decrease the weighting of the rejection model relative to the rest of the vocabulary.

If it is less than 1.0 it will reduce the probability that the rejection model is detected, and if it is more than 1.0 it will increase the probability that the rejection model is detected. Only use this if your testing reveals that the rejection model is either being detected too frequently, or not frequently enough.

It defaults to 1.0 and if you don't set it to anything (and you shouldn't, unless you have reason to believe that you should increase or decrease the probability of the rejection model being detected) it will automatically use the right setting. If you set it to a value that is equal to 1.0, or zero or less, or more than 2.0, the weight setting will be ignored and the default will be used. An NSNumber with a float looks like this: [NSNumber numberWithFloat:1.1]. The weight setting has no effect on the rest of your vocabulary, only the rejection model probabilities. Set this parameter to nil if you aren't using it.

```
- (void) deliverRejectedSpeechInHypotheses: (BOOL) trueorfalse
```

Rejecto defaults to hiding recognized statements which are not words from your vocabulary (out of vocabulary recognitions), however if you want to see them for troubleshooting purposes you can set this method to TRUE.

Note: Rejecto uses a particular token to denote phonemes that it rejects, so the delivered hypotheses will consist of these tokens, which have the format ____REJ_AA where the last part is the phoneme of the detected (and rejected) phoneme and the first part is the token which allows OpenEars and RapidEars to ignore the recognition.

Even if you are using a language model that has already been created by Rejecto and don't need to run generateRejectingLanguageModelFromArray: during your app session, it is still necessary to instantiate OELanguageModelGenerator+Rejecto and run deliverRejectedSpeechInHypotheses:FALSE if you want to see the phonemes that Rejecto is rejecting as part of your hypotheses.

If you do not need to see them, you don't have to run the method since the default is for hypotheses with Rejecto phonemes only to not be returned.