Using state of the art inexpensive audio content analysis components to build Speakularity

Or an evaluative study of automatic transcription and named entity extraction systems today

Introduction

In 2010, Nieman Labs asked journalist Matt Thompson about his pick of tools that would be introduced in 2011. He brought up the idea of Speakularity, an automatic, instant transcription system for audio and video content with a journalistic interest such as interviews, press conferences, board meetings etc.
What problems would Speakularity address? Well, there is the improving productivity and efficiency in the newsroom problem. Research shows that there can be a ratio of anywhere between 1:6 (i.e. one minute of recording requires six minutes of transcription) to 1:20. Automatic speech transcription, can, in theory reduce this time considerably. And newsrooms would not need to hire interns/contractors for the sole purpose of transcribing footage - thus freeing them to do other important tasks.
The other, and more interesting, problem that Speakularity could address is the one of searchability. Search engine technology is best at searching through text - not binary information like audio and video content. Ensuring that every spoken word out there is searchable then becomes a more manageable problem.

Extending the concept of Speakularity

While automatic transcription solves the efficiency/productivity, it only addresses searchability in a limited way. One of the ways of building the semantic web is to extract named entities from media files (at present text) and link them to other files about said entity to build an interconnected graph of files on the web. This makes the file and provides context and could then be used to build a timeline of reportage on the story, connect previously unconnected entities etc.
So Speakularity should, at the very least, be an integration of an automatic speech recognizer (ASR) and a named entity extractor (NER).

In theory yes, but...

Can we build Speakularity if we are provided with the state of the art in open source ASRs, NERs? In order to answer this question, we would need to test and compare a few of these components with each other. Also, they would need to be compared to the result of manual transcription and manual entity extraction processes.
Also worth noting here is that this is an evaluation of mostly non-commercial software. Commercial software products give purportedly better results but can be prohibitively expensive.

Test Data

We will use a recording of interviews from the American public service radio service NPR. The recording and the corresponding transcript can be downloaded from here.The advantage of using these recordings is that the quality of the recording is very high (very high signal to noise ratio), ambient sounds are minimum (improves the performance of the ASR) and the speakers use a style of intonation and prosody similar to what the ASR has been trained on.

Modifications

Recording

The original recording was an MP3 file. This file needed to be converted to a 16khz 16bit mono little-endian WAV file, as required by the ASR.

Transcript

The original transcript had several annotations such as speaker identification, music detection etc. These annotations were stripped off and the result has been used in the comparison with the output of the ASR.

Statistics

The total number of words in the transcript is 1964.
The total number of named entities is 50.

About the components

ASRs

Typically, an ASR consists of a language model, an acoustic model and a decoder.

- Language model : A statistical model trained on probabilities of sequences of words (n-grams) appearing together.
- Acoustic model : A statistical model trained on speech samples are collected, transcribed and labeled.
- Decoder : Use a modeling and scoring algorithm to find a sequence of words matching the input. This component is used to provide a score to the various hypotheses and find the best result.

CMU Sphinx

This toolkit is developed in Carnegie Mellon University and is one of the more popular open source toolkits used in speech recognition. The output is in a simple text format. The component comes with a few acoustic and language models - the most popular one is the Wall Street Journal corpus of about 5000 words. CMU Sphinx supports unigram, bigram and trigram models.

We built a demo application in CMU Sphinx which could do continuous N-gram transcribing by building a lattice for the vocabulary and which wouldn't require a separate grammar specification.

The application requires some configuration parameters such as silence insertion probability (tweaked from 0.1 to 0.9) and the depth of the N-gram model (1 = unigram, 2 = bigram etc.)

The best results were observed for a bigram model and silence insertion probability of 0.1

The output text produced with this configuration was considered as the output of the ASR in further evaluations.

NERs

An NER locates and classifies atomic elements in the input text into predefined categories which have a semantic context.

OpenCalais

This is a system developed at Thomson Reuters which can take an input text and provide entity extraction, detection and classification based on their specific training data. The system can recognize people, organizations, locations, industry terms among other entity categories. OpenCalais provides an API and a Python wrapper for invoking the API. We wrote a simple script to use the wrapper and print the results.

Stanford NER

The NER component built in Stanford University uses a statistical training model, regular expression and dictionary matching to extract entities from the input text. The component also comes built-in with these Conditional Random Field (CRF) sequence models :

- 3 classLocation, Person, Organization
- 4 classLocation, Person, Organization, Misc
- 7 classTime, Location, Organization, Person, Money, Percent, Date

The component can also be trained with other recognition models.
The tool comes with a built-in GUI (to use the tool out of the box). After loading a CRF model and entering the text, the various entities are highlighted using a color coding scheme and the output can also be exported as a tagged file.

Evaluation

Metrics

ASR

The accepted standard of evaluating ASRs is Word Error Rate (WER). Word Error Rate is defined as the Levenshtein (edit) distance between the hypothesis produced by the ASR as output and the reference (original text) texts.

Edit Distance = (Substitutions+Insertions+Deletions) / N
where N is the number of words in the original text.

If WER needs to be expressed as a percentage then, WER = Edit Distance * 100
A lower WER is desirable.

Generally, the same weight is given to all three forms of changes, but depending on the application, different weights can be assigned (like substitutions might have twice the cost of insertions and deletions). However, for the purposes of transcription, all three changes are given the same weight since they are equally critical.

The NISTAlign class is an in-built utility class in Sphinx which can align the reference and hypotheses texts and subsequently calculate WER and accuracy.

NER

According to the Message Understanding Conference (MUC) website, the metric used for evaluating NER systems is as follows :

Given a set of points, there are several values calculated in the alignment and final scoring.
COR (Correct) : The number of correct points
INC (Incorrect) : The number of incorrect points
MIS (Missing) : The number of missing points

SPU (Spurious) : The number of spurious points

Then,
POS (Possible) : The number of points from single fill mappings which contained a reference single fill.
POS = COR + INC + MIS

ACT (Actual) : The number of points from single fill mappings which contained a hypothesis single fill.
ACT = COR + INC + SPU

REC (Recall) : a measure of how many of the reference fills were produced in the hypothesis.
REC = COR / POS

PRE (Precision) : a measure of how many of the hypothesis fills are actually in the reference.
PRE = COR/ACT

F-measure
a function used to combine recall and precision measures into one measure [VANRIJSBERGEN] . The formula for F is

$$F = \frac{((\text{beta})^2 + 1.0) * PRE * REC}{((\text{beta})^2 * PRE) + REC}$$

where beta is the relative weight of precision and recall. When precision and recall are given equal weight, the value for beta is 1.

Substituting 1 for beta, and the previous formulas for precision and recall, the above formula simplifies to

F = (2 * COR) / (POS + ACT)

The definition of named entity categories can be contextual, hence there is no standard. However, the BBN catalog of named entity categories has been considered here as a reference. Hence missing entities etc. have been listed based on this catalog.

Evaluation scenarios

The research question can be best answered by the evaluation of the following four scenarios:

Manual transcription - manual entity extraction

Manual transcription

The manual transcription (created by the NPR freelancer) had

Substitutions = 0.2% (5)

Deletions = 0.35% (7)

Insertions ≈ 0% (1)

Errors = 0.66% (13)

WER = (13 * 100)/1964 = 0.66%

Manual entity extraction

COR (Correct) = 50

INC (Incorrect) = 0

MIS (Missing) = 0

SPU (Spurious) = 0

POS = COR + INC + MIS = 50

ACT = COR + INC + SPU = 50

REC = COR / POS = 50/50

PRE = COR/ACT = 50/50

F = (2 * COR) / (POS + ACT) = (2*50) / (50 + 50) = 1

Manual transcription - automated entity extraction

Manual transcription

The manual transcription (created by the NPR freelancer) had

Substitutions = 0.2% (5)

Deletions = 0.35% (7)

Insertions ≈ 0% (1)

Errors = 0.66% (13)

WER = (13 * 100)/1964 = 0.66%

Automated entity extraction using Stanford NER

COR (Correct) = 31

INC (Incorrect) = 0

MIS (Missing) = 19

SPU (Spurious) = 7

POS = COR + INC + MIS = 31 + 19 = 50

ACT = COR + INC + SPU = 31 + 7 = 38

REC = COR / POS = 31/50

PRE = COR/ACT = 31/38

F = (2 * COR) / (POS + ACT) = (2*31) / (50 + 38) = 62/88 = 0.70

Automated entity extraction using OpenCalais

COR (Correct) = 34

INC (Incorrect) = 1

MIS (Missing) = 16

SPU (Spurious) = 3

POS = COR + INC + MIS = 34 + 16 = 50

ACT = COR + INC + SPU = 34 + 1 + 3 = 38

REC = COR / POS = 34/50

PRE = COR/ACT = 34/38

F = (2 * COR) / (POS + ACT) = (2*34) / (50 + 38) = 68/88 = 0.77

Automated transcription - manual entity extraction

Automatic transcription

Substitutions = 74.7% (1465)

Deletions = 2.2% (43)

Insertions = 35.6% (697)

Errors = 112.5% (2205)

WER = (2205 * 100)/ 1960 = 112.500%

Manual entity extraction

COR (Correct) = 3

INC (Incorrect) = 0

MIS (Missing) = 0

SPU (Spurious) = 0

POS = COR + INC + MIS = 3

ACT = COR + INC + SPU = 3

REC = COR / POS = 3/3

PRE = COR/ACT = 3/3

F = (2 * COR) / (POS + ACT) = (2*3) / (3 + 3) = 1

Automated transcription - automated entity extraction

Automatic transcription

Substitutions = 74.7% (1465)

Deletions = 2.2% (43)

Insertions = 35.6% (697)

Errors = 112.5% (2205)

WER = (2205 * 100)/ 1960 = 112.500%

Automated entity extraction using Stanford NER

COR (Correct) = 1

INC (Incorrect) = 0

MIS (Missing) = 2

SPU (Spurious) = 1

POS = COR + INC + MIS = 1 + 2 = 3

ACT = COR + INC + SPU = 1 + 1 = 2

REC = COR / POS = 1/3

PRE = COR/ACT = 1/2

F = (2 * COR) / (POS + ACT) = (2*1) / (3 + 2) = 2/5 = 0.4

Automated entity extraction using OpenCalais

COR (Correct) = 2

INC (Incorrect) = 0

MIS (Missing) = 1

SPU (Spurious) =

POS = COR + INC + MIS = 2 + 1 = 3

ACT = COR + INC + SPU = 2+ 0 + 0 = 2

REC = COR / POS = 2/3

PRE = COR/ACT = 2/2

F = (2 * COR) / (POS + ACT) = (2*2) / (3 + 2) = 4/5 = 0.8

Summary

Scenario	**WER**	**F-measure**
Manual transcription - manual entity extraction	0.66%	1

Manual transcription - automated entity extraction	0.66%	Stanford NER : 0.70 Open Calais : 0.77
Automated transcription - manual entity extraction	112.5%	1
Automated transcription - automated entity extraction	112.5%	Stanford NER : 0.4 Open Calais : 0.8

Findings

The scenario that produced the best results is where the transcription was done manually, as was the entity extraction. Nonetheless, the results from the scenario where the transcription was done manually and the entity extraction was automated were also comparable to those of the first scenario.

This implies that at the very least, the entity extraction module can be used in conjunction with a content management system / interview storage database to provide appropriate tags so that the recording is indexable and searchable within the repository. It also helps to provide semantic tagging to the document since the entities extracted were very relevant to the context of the story. It is also a faster process than manual entity extraction/

So, the recommended scenario is Manual transcription - automated entity extraction

Improvements

Although the WER of the CMU Sphinx is high, there are ways to improve it namely increasing the acoustic and language models. At the moment, the available vocabulary is limited, so a vocabulary set like the English Gigaword set or Google n-gram might be more applicable. In order to adapt the acoustic model, we would need to create a list of sentences, a dictionary describing the pronunciation of all the words in that list of sentences, and a recording of you speaking each of those sentences. However, since the speaker and the content is always different, the acoustic model would need to constantly be adapted. This is time-consuming and processing-intensive but it can be done.

Further, integrating a speaker identification system, silence and music identification system will also help improve the transcription significantly.
It is also worth investigating the Google Voice API once it is released - Google's ASR technology is state of the art as evidenced by its voice search feature.

Conclusion

Automatic speech recognition is analogous to an NP-complete problem. But current speech recognition technology is constantly under improvement and can produce desired results if it is augmented with the appropriate acoustic and language models. This would imply that a significant amount of time and effort would be needed to have a basic speech recognizer for interview transcriptions.
Named entity recognition can definitely assist reporters in cataloging their interview files to make them better searchable, indexable and to add context to the interview by tagging it appropriately.

So, we are not quite at that stage where Speakularity can be used commercially and considered reliable. But with several improvements and constantly updating the training data, it is possible to build a version that could produce desirable transcription output efficiently and reliably.