Speech-to-Text with SAS Viya

Philip Sierpinski
Solution Advisor, AI & Advanced Analytics

What is Speech-to-text (STT) ?

How does it work?

Why is STT used?

What is Speech-to-text (STT) ?

Speech to text is a speech recognition software that enables the recognition and translation of spoken language into text through computational linguistics. It is also known as speech recognition or computer speech recognition.

What is Speech-to-text (STT) ?

How does it work?

Why is STT used?

Why is STT used?

How sentiment and topics can be used to improve customer experience and prevent their problems

From Data

...through Analytics

...to Improved customer experience

Transcription
Emails
Chat Logs

Sentiment

Topics

- Identify customers requiring retention
- Identify agents requiring training for handling difficult customers
- Augment campaign decisioning with sentiment, choosing the right NBA – retain or sell?

- Identify most common customer problems and adjust the process to avoid them – prevention
- Identify customer needs and propose product / solution accordingly

Other applications:

Automatic subtitles

Virtual assistants (Siri, Alexa, Google home)

Aerospace

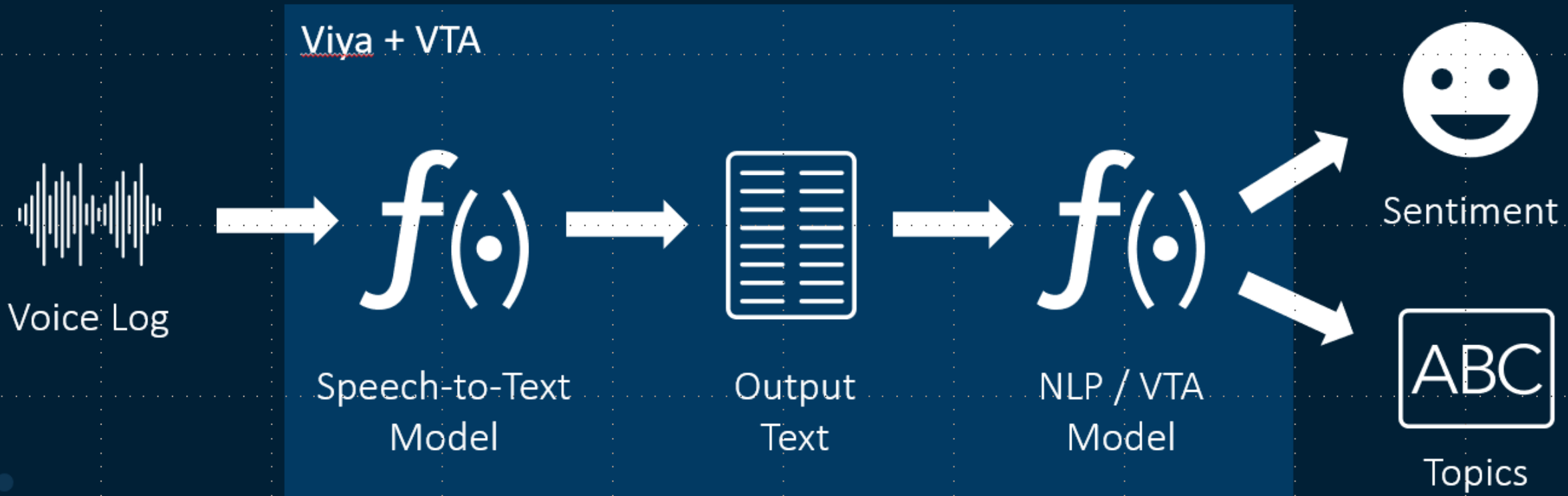
What is Speech-to-text (STT) ?

How does it work?

Why is STT used?

How does it work?

High Level Overview



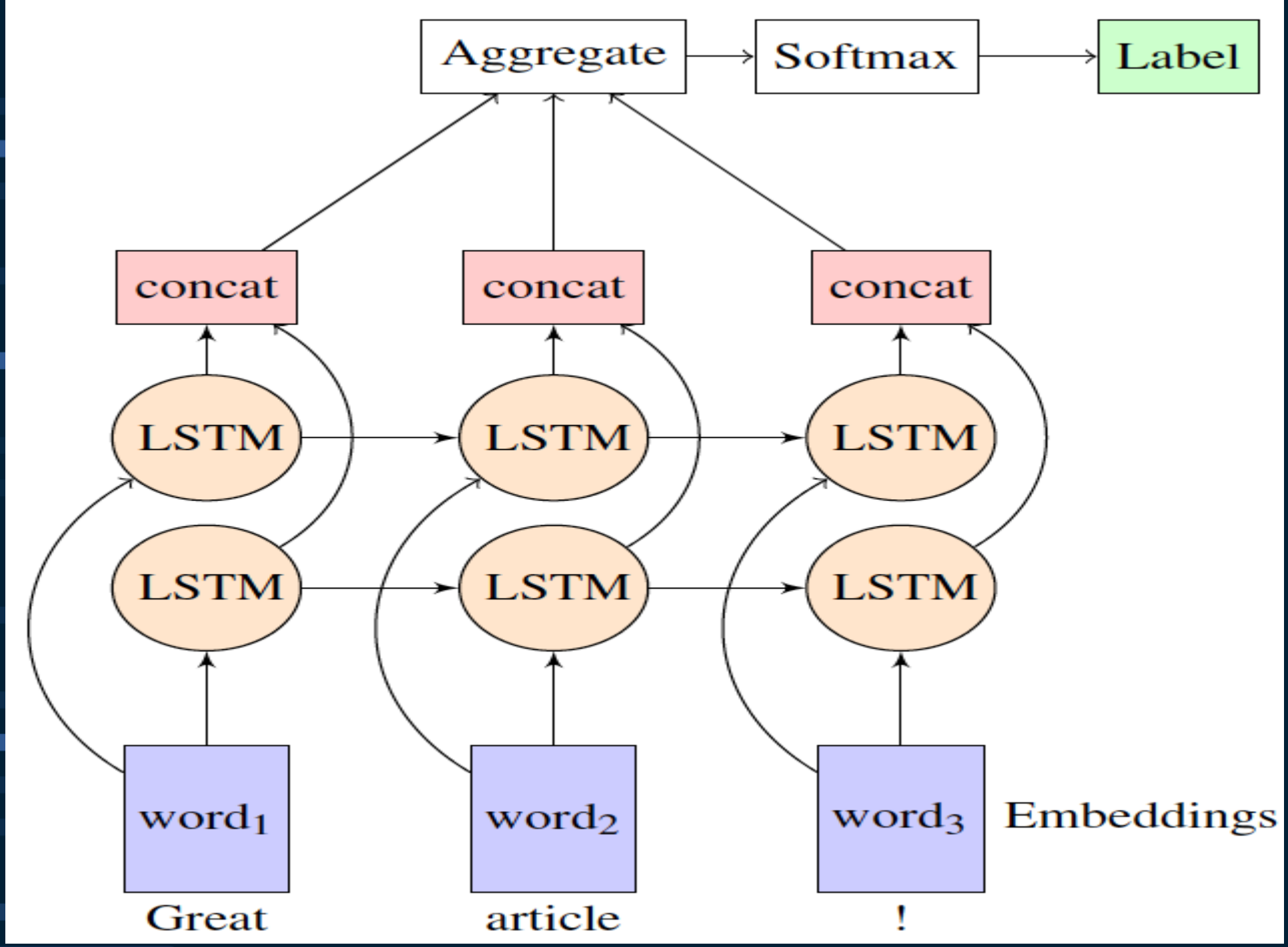
2 parts of the STT pipeline:

Acoustic model

Language model

Acoustic model

- Deals with the raw audio waveforms of human speech
- Encodes all possible characters


Aggregate
Softmax
Label
concat
concat
concat
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
word1
word2
word3
Embeddings
Great
article
!

Language model

- Creates the structure for the acoustic model by decoding the predicted characters (by the acoustic model) into words
- Can be used across several different STT projects, with different acoustic models trained on separate data sets

Language model

Output from acoustic model

[J,A,G,J,O,B,B,A,R,P,Å,S,A,S,O,C,H,H,E,T,E,R,P,H,I,L,I,P]

Output from Language model

“JAG JOBBAR PÅ SAS OCH HETER PHILIP”

Required Library Packages to Load into Python

```
import os
import sys

from swat import *

import getpass
```

Connecting and Setting up SAS Viya Environment

```
os.environ['CAS_CLIENT_SSL_CA_LIST'] = r'<cert location>'

s = CAS("<server>", port=<port>, username='<log in username>', password=getpass.getpass("Password: "))

s.loadactionset("audio")
s.loadactionset("searchanalytics")
s.loadactionset("deeplearn")
s.loadactionset("langmodel")
```

```
s.addcaslib(name="myCasLib",
            path="<path to files>",
            activeOnAdd = True,
            dataSource={"srctype":"path"})

s.addcaslib(name="audioCasLib",
            path="<path to audio>",
            activeOnAdd = False,
            dataSource={"srctype":"path"})
```

Import Speech to Text Model (Acoustic and Language Models)

```
s.langmodel.lmImport(table="language_model.sashdat",
                     casout={"name":"og_lm",
                             "caslib":"myCasLib",
                             "replace":True})
```

```
s.table.loadTable(path="acoustic_model_cpu.sashdat",
                  caslib="myCasLib",
                  casout={"name":"asr",
                          "caslib":"myCasLib",
                          "replace":True})

s.table.loadTable(path="acoustic_model_cpu_weights.sashdat",
                  caslib="myCasLib",
                  casout={"name":"pretrained_weights",
                          "caslib":"myCasLib",
                          "replace":True})

s.table.loadTable(path="acoustic_model_cpu_weights_attr.sashdat",
                  caslib="myCasLib",
                  casout={"name":"pretrained_weights_attr",
                          "caslib":"myCasLib",
                          "replace":True})

s.table.attribute(task="ADD",
                  name="pretrained_weights",
                  attrtable="pretrained_weights_attr")
```

Loading Input Data

```
s.audio.loadAudio(path = "audio.txt",
                  caslib = "audioCasLib",
                  casout = {"name":"audio",
                            "caslib":"myCasLib",
                            "replace":True})

vars_list = ["_path_"]

nFrames = 3500
nToken = 40

s.audio.computeFeatures(table = "audio",
                        copyVars = vars_list,
                        casout = {"name":"scoring_data",
                                  "caslib":"myCasLib",
                                  "replace":True},
                        audioColumn = "_audio_",
                        frameExtractionOptions = {"frameshift":10,
                                                  "framelength":25,
                                                  "dither":0.0},
                        melBanksOptions = {"nBins":nToken},
                        mfccOptions = {"nCeps":nToken},
                        featureScalingMethod = "STANDARDIZATION",
                        nOutputFrames = nFrames)
```

Scoring with Acoustic Model

```
s.dlscore(table = "scoring_data",
          modelTable = "asr",
          initWeights = "pretrained_weights",
          nThreads = 12,
          copyVars = ["_path_"],
          casout = {"name":"scoredData", "replace": True})
```

Decode Scored Data with Language Model

```
s.langModel.lmDecode(table = "scoredData",
                     langModelTable = "og_lm",
                     copyVars = ["_path_"],
                     blankLabel = " ",
                     spaceLabel = "&",
                     casout = {"name":"results",
                               "replace": True})
```

Calculate Error Rates

To calculate the error rate, the original transcripts of the audio files have to be first cleaned and loaded as a single file, as done so by the code snippet below.

```
wav_folder = "<input directory>"
audio_folder = "<myCasLib directory>"
file_out = open(wav_folder + "actual_transcription.txt", "w")
file_out.writelines("_path_,_transcript_\n")

for f in os.listdir(audio_folder):
    if f.endswith(".trans.txt"):
        file_in = open(audio_folder + f, "r") # open the file to read
        txt_flist = file_in.readlines()
        file_in.close()

        trans_list = []
        for line in txt_flist:
            line = line.strip()
            # find(): finds the first occurence of a specified string
            audio_id = wav_folder + line[:line.find(" ")] + ".wav" # finds the first space and adds ".wav" to the text before the first space
            audio_text = line[line.find(" ") + 1:] # reads from the first character after the space within the line
            trans = audio_id + "," + audio_text.strip() + "\n"
            trans_list.append(trans)


        trans_list.sort()
        file_out.writelines(trans_list)

file_out.close()

s.table.loadtable(path="actual_transcription.txt",
                  caslib="myCasLib",
                  casout={"name":"reference_table",
                          "caslib":"myCasLib",
                          "replace":True})
```

```
s.calculateErrorRate(table={"name":"results",
                            "caslib":"myCasLib"},
                     tableId="_path_",
                     tableText="_audio_content_",
                     reference={"name":"reference_table",
                                "caslib":"myCasLib"},
                     referenceId="_path_",
                     referenceText="_transcript_")
```

Questions?