Speech Recognition

- With Artificial Neural Networks -

Project Report

ED5-1-E17

Aalborg University
Electronics and IT

Here you can write something about which tools and software you have used for typesetting the document, running simulations and creating figures. If you do not know what to write, either leave this page blank or have a look at the colophon in some of your books.

Electronics and IT
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Speech Recognition

Theme:
Scientific Theme

Project Period:
Fall Semester 2017

Project Group:
ED5-1-E17

Participant(s):
Stefan Bîrs
Tiberiu-Ioan Szatmari
Kamran Thomas Alimagham

Supervisor(s):
Daniel Ortiz-Arroyo
D. M. Akbar Hussain

Copies: 1

Page Numbers: 29

Date of Completion:
November 16, 2017

Abstract:

This report goes into detail about the implementation of distributed speech recognition systems, information retrieval and machine learning. The main objective of the speech recognition system is to provide a handsfree human-computer interface experience, where the system is also able to handle noise.

Contents

Todo list

Preface

The project entitled Speech Recognition with Focus on information retrieval and command prompt was made by three students from the Electronics and Computer Engineering programme at Aalborg University Esbjerg, for the P5 project during the fifth semester.

Aalborg University, November 16, 2017

Stefan Bîrs
<sbars15@student.aau.dk>

Tiberiu-Ioan Szatmari
<tszatm15@student.aau.dk>

Kamran Thomas Alimagham
<kalima15@student.aau.dk>

Chapter 1

Introduction

The focus of this project will be on implementing speech recognition system...

The main focus is to design and implement an different artifitial neural networks (ANN) and try different regulation techniques to get the best performing speech recognition system we can achieve...

The particular software platform is the open source TensorFlow [1] software library released by Google, where it allows non-AI (Artificial Intellegent) experts to design, build, and train deep learning models. In order to create a control solution for the said speech recognizer it is necessary to create a artifitial neural network...

Speech is probably the most important forms of communication. Because its the most natural and efficient way to communicate with each other. Humans learn all the relevant skills during early childhood, without any instruction, and they continue to rely on speech communication throughout their life [2]. Humans also want to have a similar natural, easy and efficient form of communication with machines. Until recently, the idea of holding a conversation with a computer seemed pure science fiction.

But things are changing, and quickly. A growing number of people now talk to their mobile smart phones, asking them to set reminders, search for directions, or send email and text messages. Chief technology officer of © Nuance Communications Vlad Sejnoha stated, "We're at a transition point where voice and natural-language understanding are suddenly at the forefront".

[2]

Chapter 2

Problem Description

2.1 Problem Description

List to wright about here:

- Main Objective/Goal
- What will be required
- How are we going solve the Objective?

2.2 Problem Delimitation

To ensure that the project meets the desired scope, specific success criteria are formed which will be evaluated during the final stages of the report. The following list of requirements are the desired success criteria, where the focus will ve on reaching the given goals.

- Detect human speech
- Design a deep neural network
- Compare different regulation technics (drop out, bach normalization...)
- Design a filter
- Create a network infrastructure for Client/Server Model (Distribution)
- Implement a speech recognition system with information retrival

Brief summary on how are we going to obtain all of these bullet points.

Chapter 3

Speech Processing

Speech is an effortless and highly efficient form of communication. Continuous speech recognition software is readily available from stores and it allows the user to interact with its surrounding without the need of written commands [3, p. 396]. Amazon's Alexa [4] and Microsoft's Cortana [5] are perfect examples of home companions that are able to perform basic tasks received via voice commands. As speech recognition increases in accuracy to the high nineties, reaching a performance of 99% means that this technology will make the leap from being annoying to use to becoming the main way we interact with computers. This idea is further backed up by the fact that people tend to speak much faster than they are able to type [6]. In some cases, speech can be three times faster in conveying information than typing and for professional debaters, speaking can be even six times faster.

Words are carried as sound waves, which are analog signals. In order for the speech to be processed, it needs to be run through a signal processor that selects the frequencies and amplitude of the signal. Further on, the signal is mapped to individual sound units called phones. These phones need to be identified and grouped in such a way to ensure that each word has a different phonetic structure, otherwise, words would be impossible to distinguish from one another.

Table 3.1: Phones

[ay]	iris
[b]	bin
[er]	bird
[l]	lip
[p]	pin
[th]	thin

Phones can have different sounds depending on the context. For example, the phone th in the word *three* has a different sound to th in *then*. To overcome these different variations of the same phones, it is better to abstract the phones into a generalized grouping called a **phoneme**. Phonemes are written in-between forward slashes (*/th/*) and they will have a specific pronunciation for each sound, depending on the context. These phonemes are used as a transitional layer when trying to convert speech to text and vice versa, when speech needs to be synthesized.

3.1 Signal processing

Sound waves that carry human speech are variations in air pressure. The key components of a sound wave are its **amplitude**, which measure the intensity of the sound and the **frequency**, which describes the rate at which the amplitude varies over time. When speaking in a microphone, the change in air pressure causes the diaphragm to oscillate. The size of the oscillations is directly proportional to the amplitude of the signal, while the rate at which the diaphragm oscillates gives represents the rate at which the air pressure changes. At specific time intervals, the signal can be sampled and the data can be used in a wide array of digital signal processing tools. For example, the digital signal can be plotted as an x-y plot, where the y-axis defines the amplitude and the x-axis shows the time. The frequency can be easily be determined from the plot as we find the number of cycles that the signal does per second.

Although a visual difference can be seen when plotting vowels and consonants, a visual inspection of the waveform will not reveal any critical information, making it impossible to see all the phonemes.

To feed this information to the neural network, some preprocessing and signal manipulation is required so the input can be understood and passed through the neurons. These steps are expanded upon in the following subsections.

Preprocessing

No matter how powerful and how advanced a computer is, it still works in discrete time. In order to represent an analogue signal on the computer we have to sample it according to the Nyquist–Shannon sampling theorem, which states that the sampling frequency has to be at least double the frequency of the original signal in order to replicate the original signal without aliasing. For a better understanding of the process, a visual representation of the word "Hello" as it is formed for the

input of the neural network is shown in figure**??** where it is sampled at 8000 Hz.

Fourier transform

To analyse a discrete signal we use the Fourier transform. This allows us to convert the signal in the frequency domain and deconstruct it in a series of simple sine waves. These sine waves have specific amplitudes and frequencies and while plotting the FFT of the signal, the dominant waves can be identified by the intensity of the amplitude.

In Figure **??**, a visual representation of the FFT is shown. It can be seen that there are a lot of frequencies that make up the word "Hello". The dominant ones are in the range of 500Hz. Considering that this recording has a minimum amount of noise, we have to consider the full range of the frequencies, from zero and up to 3500Hz.

Spectrogram

The spectrogram of the signal was computed from the FFT and it shows a visual representation of the spectrum of frequencies marked by the intensity of the color. After the spectrogram is sliced into 25 ms chunks, it is feed as the input to the network. A neural network can find patterns in this data more easily than raw sound waves because audio data and patterns can be seen in the graph.

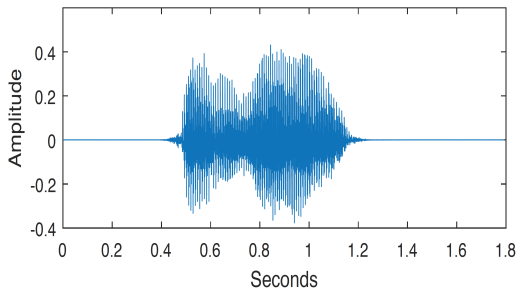

Figure 3.1: Left caption.

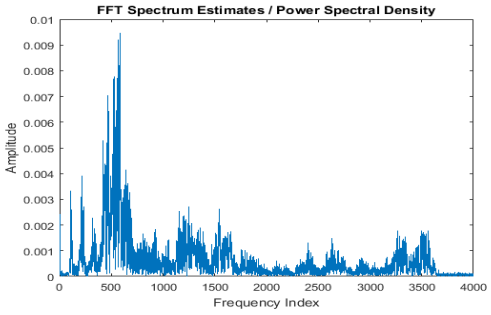

Figure 3.2: Right caption.

The figures are on GitHub, need to figure out how they work in LaTeX

Chapter 4

Machine Learning for Speech

4.1 Artificial Neural Networks

4.1.1 What is an Artificial Neural Network?

Artificial Neural Networks are nothing but a computerized representation of the human brain. The human brain learns from its experiences, creating new neural pathways and adopts new habits by increasing or decreasing certain neural interconnections. This brain modeling approach is a less technical way to develop machine learning solutions, but it is known to be more efficient and accurate that previous traditional approaches, such as the Hidden Markov Model (HMM).

4.1.2 Artificial Neuron

The term "Neural" is derived from the basic funtional unit "Neuron" which typically comprises of these four parts:

1. Dendrites: Accept inputs
2. Soma: Process the inputs
3. Axon: Turn the processed inputs into outputs
4. Synapses: The electrochemical connections between neurons

The connections can be inhibitory (decreasing strength) or excitory (increasing strength) in nature.

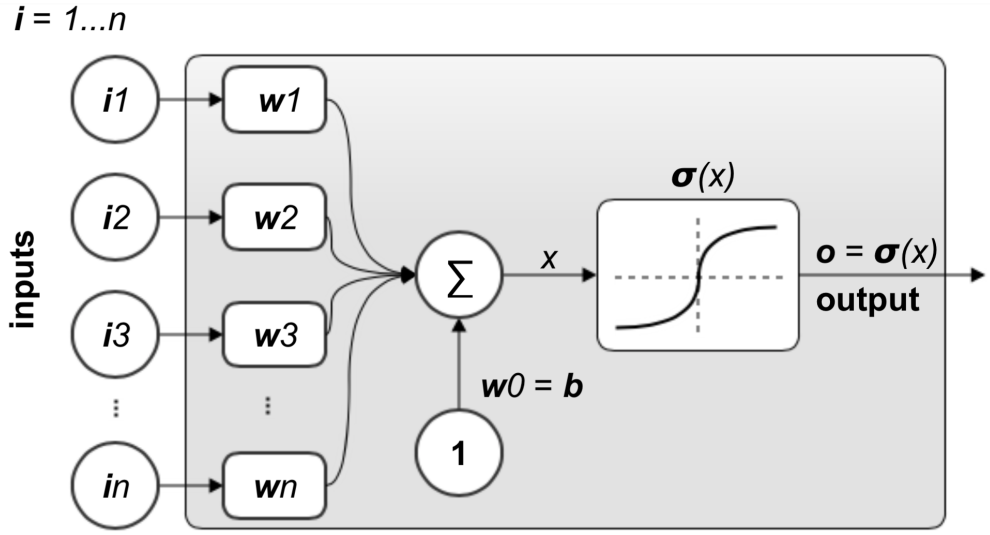

Figure 4.1: Basic Artificial Neuron.

4.2 Types of Artificial Neural Networks

4.2.1 Feedforward Network

This type of simple neural network is comprised of one input layer and one neural layer, which also acts as the output layer. The flow of information is uni directional, starting from the input layer and progressing to the output layer. As such, the number of outputs from the neural network will always coincide with the number of neurons in the network. These networks are usually employed in pattern classification and linear filtering problems. The Perceptron and the ADALINE architectures are among the most recognised feed-forward neural networks and they work well with Hebb's rule and the Delta rule for training.

4.2.2 Deep Recurrent Neural Network

To explain how a deep recurrent neural network looks like first we will see what a deep neural network is made of and after that the recurrent part will be added to form a more complex and and powerful network.

Networks with multiple layers are have one or more hidden neural layers. Such a network will always have one input layer with multiple samples and a series of variable layers. In contrast to the simple feed-forward network, these hidden layers are not constrained to have the same size as the input layer. This means that the size can depend on the complexity of the problem being tackled by the network, as

well as the quantity and quality of the available data. The last layer still occupies a double role as both a neuron layer and output layer.

Another key element is the fact that the outputs of the neurons are used for feedback inputs to the other neurons. This feature makes the network great for time-variant systems where the new outputs can benefit from previous information.

4.2.3 Long Short Term Memory

Long Short Term Memory networks, for short "LSTMs", are a special case of recurrent neural network that is capable of learning long-term dependencies. Firstly used in 1997 by Sepp Hochreiter and Jürgen Schmidhuber [7] and refined by Felix Gers and his team [**Gers99**]. Nowadays, after further refinement, LSTMs work incredibly well on various problems and give better results for most problems that require RNNs. They remember information for long periods of time by default, making them the perfect fit to avoid long-term dependency problems.

LSTMs have a chain like structure, with repeating modules and layers interacting in a specific way.

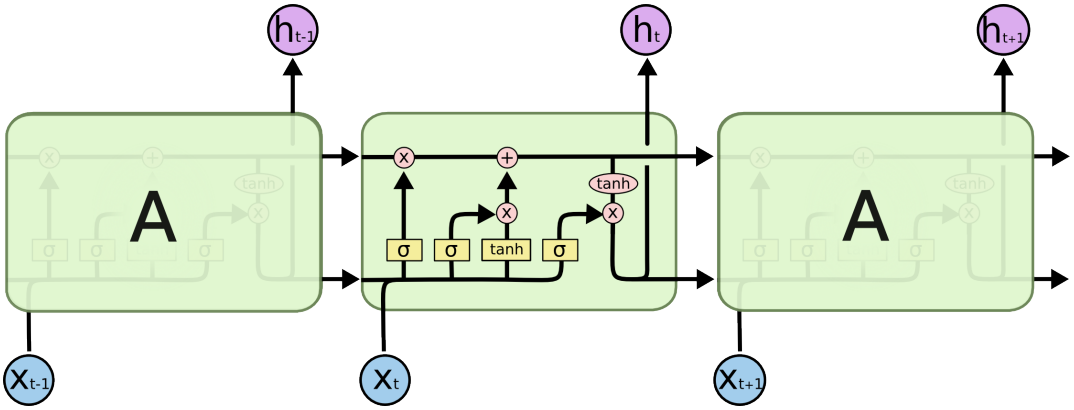

Figure 4.2: LSTM Diagram.

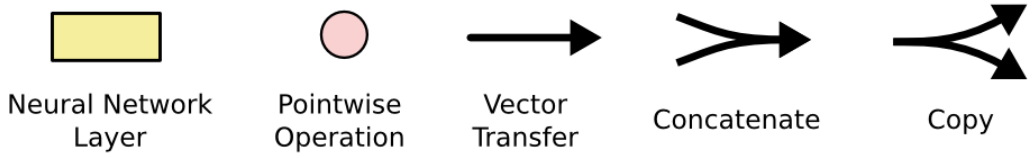

Figure 4.3: Diagram legend.

In the above diagram, each line carries an entire vector, from the output of one node to the inputs of others. The pink circles represent pointwise operations, like

vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation, while a line forking denote its content being copied and the copies going to different locations.

4.3 TensorFlow

TensorFlow provides multiple Application Programming Interfaces (APIs) for machine learning. The lowest level API "TensorFlow Core" provides complete programming control [1]. For those who require fine levels of control over their models, TensorFlow Core is a well-suited tool for the job. There are higher level APIs that are built on top of TensorFlow Core. These higher level APIs are typically easier to learn and use than Tensorflow Core [1]. In addition, the higher level APIs such as "tf.estimator" helps with managing data sets, estimators, training and inferences (testing your trained network), as well as, making repetitive tasks easier and more consistent [1].

The subsection below will start with TensorFlow Core. In order to gain an understanding of the basic principles that TensorFlow has to offer, a model shall be made. In the subsection after that, the same model will be implemented with tf.estimator. Knowing TensorFlow Core principles will give us a great mental model of how things are working internally when we use the more compact higher level API.

4.3.1 Tensors

The central unit of data in TensorFlow is the tensor. A tensor consists of a set of primitive values shaped into an array of any number of dimensions. A tensor's rank is its number of dimensions. Here are some examples of tensors:

```
3 # a rank 0 tensor; this is a scalar with shape []
[1., 2., 3.] # a rank 1 tensor; this is a vector with shape [3]
[[1., 2., 3.], [4., 5., 6.]] # a rank 2 tensor; a matrix with shape [2, 3]
[[[1., 2., 3.]], [[7., 8., 9.]]] # a rank 3 tensor with shape [2, 1, 3]
```

customize colors

4.3.2 TensorFlow Core

Getting Started With TensorFlow is done by using the import statement. This gives Python access to all of TensorFlow's classes, methods, and symbols and is called by the following statement:

```
import tensorflow as tf
```

All the programs written with the help of TensorFlow core are built around computational graphs. They are a series of operations arranged into a graph and

connected by nodes. To see the outcome of a such a program, firstly the computational graph needs to be created and after that it needs to be run. This provides a contrast to normal code written in python and to display this, the print function for a tensorflow constant is called bellow.

```
node1 = tf.constant(3.0, dtype=tf.float32)
node2 = tf.constant(4.0) # also tf.float32 implicitly
print(node1, node2)

Output:
Tensor("Const:0", shape=(), dtype=float32)
Tensor("Const_1:0", shape=(), dtype=float32)
```

As seen above, printing an element does not show the value that it holds and it rather shows the technical details behind, such as the element being a constant of type float32. To investigate the contents of any element in tensorflow, an object of type session needs to be defined. When the new object is run, a new session is generated and the content of the element can be seen. All code written is tensorflow core follows this basic principle.

```
sess = tf.Session()
print(sess.run([node1, node2]))

Output:
[3.0]
```

Other basic elements in tensorflow are placeholders(tf.placeholder), they can be changed to accept external inputs and variables. Variables allow the system to change its outputs while keeping the seame inputs, this allows the model to be trainable.

4.3.3 tf.train API

The tf.train API holds optimizers that change each variable to minimize the loss function. One of the simplest optimizers used to train neural networks is gradient descent. Each variable is modified by the magnitude of the derivative of loss with respect to that variable. Using a computer to gain these gradients is far less prone to error than doing it by hand. Each of the gradients will point towards the minimum of the loss function and when solving for the gradient a step needs to be determined as the rate of change. It is important to have a small step as to not jump over the valley where the function takes its minimum value, but in the beginning to reduce time and processing power a bigger step could be used. One solution to this problem is to use a variable step input that changes as the gradients are determined. TensorFlow can automatically produce derivatives given only a description

of the model using the function tf.gradients. For simplicity, optimizers typically do this for us.

```
optimizer = tf.train.GradientDescentOptimizer(0.01)
train = optimizer.minimize(loss)
sess.run(train)
```

The gradient descent optimizer takes an input, defined at 0.01 in the example above, that represents the step input or the rate of change for the gradients.

4.3.4 tf.estimator API

To simplify the procedure of machine learning, tf.estimator can be used as a higher-level library within Tensorflow. It helps the user with running training and evaluation loops, managing data sets and much more. Making the entire process of writing an algorithm and maintaining it be less tedious. Although the estimator library has a set of predefined models to make things easier, a custom model can be created while keeping the high level abstraction of the data set, training and feeding. A linear regression algorithm built with tf.estimator is included below to show how the library works [8]:

```
import numpy as np
import tensorflow as tf

# Declare list of features. We only have one numeric feature. There are m
# other types of columns that are more complicated and useful.
feature_columns = [tf.feature_column.numeric_column("x", shape=[1])]

# An estimator is the front end to invoke training (fitting) and evaluatio
# (inference). There are many predefined types like linear regression,
# linear classification, and many neural network classifiers and regresso

# The following code provides an estimator that does linear regression.
estimator = tf.estimator.LinearRegressor(feature_columns=feature_columns)

# TensorFlow provides many helper methods to read and set up data sets.
# Here we use two data sets: one for training and one for evaluation
# We have to tell the function how many batches
# of data (num_epochs) we want and how big each batch should be.
x_train = np.array([1., 2., 3., 4.])
y_train = np.array([0., -1., -2., -3.])
x_eval = np.array([2., 5., 8., 1.])
y_eval = np.array([-1.01, -4.1, -7, 0.])
input_fn = tf.estimator.inputs.numpy_input_fn(
```

```
    {"x": x_train}, y_train, batch_size=4, num_epochs=None, shuffle=True)
train_input_fn = tf.estimator.inputs.numpy_input_fn(
    {"x": x_train}, y_train, batch_size=4, num_epochs=1000, shuffle=False)
eval_input_fn = tf.estimator.inputs.numpy_input_fn(
    {"x": x_eval}, y_eval, batch_size=4, num_epochs=1000, shuffle=False)

# We can invoke 1000 training steps by invoking the  method and passing the
# training data set.
estimator.train(input_fn=input_fn, steps=1000)

# Here we evaluate how well our model did.
train_metrics = estimator.evaluate(input_fn=train_input_fn)
eval_metrics = estimator.evaluate(input_fn=eval_input_fn)
print("train metrics: %r"% train_metrics)
print("eval metrics: %r"% eval_metrics)
```

4.4 Deep neural network for speech

4.4.1 Deep Feed Forward (DFF)

The simplest of all neural networks, the feedforward neural network, moves information in one direction only. Data moves from the input nodes to the output nodes, passing through hidden nodes (if any). The feedforward neural network has no cycles or loops in its network.

reference this part with *https://www.allerin.com/blog/six-types-of-neural-networks*

4.4.2 Recurrent Neural Nerwork (RNN)

The recurrent neural network, unlike the feedforward neural network, is a neural network that allows for a bi-directional flow of data. The network between the connected units forms a directed cycle. Such a network allows for dynamic temporal behavior to be exhibited. The recurrent neural network is capable of using its internal memory to process arbitrary sequence of inputs. This neural network is a popular choice for tasks such as handwriting and speech recognition.

link this part with link above

Long/Short Term Memory (LSTM)

Chapter 5

Distributed speech recognizer

5.1 The DSR system

5.2 Acoutic models

5.3 Language model

Chapter 6

Knowledge-based IR system

6.1 The document retrieval system

6.2 The question answering system

Chapter 7

Discussion

Chapter 8

Conclusion

In case you have questions, comments, suggestions or have found a bug, please do not hesitate to contact me. You can find my contact details below.

Jesper Kjær Nielsen
jkn@es.aau.dk
http://kom.aau.dk/~jkn
Fredrik Bajers Vej 7
9220 Aalborg Ø

Bibliography

[1] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[2] Bhushan C Kamble. "Speech Recognition Using Artificial Neural Network–A Review". In: ().

[3] R. Callan. *Artificial Intelligence*. Macmillan Education UK, 2003. ISBN: 9780333801369. URL: https://books.google.dk/books?id=S1N0QgAACAAJ.

[4] Darrell Etherington. *Alexa*. https://techcrunch.com/2014/11/06/amazon-echo/. Accessed: 2017-10-30.

[5] *Cortana*. https://support.microsoft.com/en-us/help/17214. Accessed: 2017-10-30.

[6] Arie Ogranovich. *Speed*. https://www.quora.com/Can-we-type-in-English-faster-than-we-speak. Accessed: 2017-10-30.

[7] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: 9 (Dec. 1997), pp. 1735–80.

[8] https://www.tensorflow.org/get_started/get_started. Accessed: 2017-11-15.

Appendix A

Appendix A name

Here is the first appendix