



UNIVERSITEIT•STELLENBOSCH•UNIVERSITY  
jou kennisvennoot • your knowledge partner

# **Neural networks for language model smoothing**

Werner Van der Merwe  
20076223

Report submitted in partial fulfilment of the requirements of the module  
Project (E) 448 for the degree Baccalaureus in Engineering in the Department of  
Electrical and Electronic Engineering at Stellenbosch University.

Supervisor: Prof. T. R. Niesler

November 2019

# Acknowledgements

...

# Declaration

I, the undersigned, hereby declare that the work contained in this report is my own original work unless otherwise stated.

Signature: .....  
Werner Van der Merwe

Date: .....

# Abstract

## English

...

” An automatic speech recogniser uses an internal statistical model of plausible word sequences (the language model) to guide its search during speech-to-text transcription. A classic but still popular choice of language model is the so-called n-gram, which uses the relative frequencies of word n-tuples in a training set to estimate the probabilities of entire word sequences. Even with the huge text databases available today, however, many legitimate n-tuples are never seen during training. For these situations the language model must make use of smoothing to estimate the probability of such unseen events. Successful smoothing approaches include the backoff, the Good-Turing estimator and especially the so-called Kneser-Ney (KN) estimate (named after the proposers). The goal of this project is to see whether a neural network can be trained to deliver the required smoothing instead of a predetermined algorithm like KN smoothing, and to determine how neural smoothing compares with established techniques. This project requires excellent programming skills, the ability to function in a Linux-only environment, and strong computational skills (mathematics and statistics). ”

# Contents

<b>Declaration</b>	<b>ii</b>
<b>Abstract</b>	<b>iii</b>
<b>List of Figures</b>	<b>v</b>
<b>List of Tables</b>	<b>vi</b>
<b>Nomenclature</b>	<b>vii</b>
<b>1. Introduction</b>	<b>2</b>
<b>2. Literature Study</b>	<b>3</b>
2.1. The Language Model . . . . .	3
2.1.1. The Need for the Statistical Language Model . . . . .	3
2.1.2. Math behind the Model . . . . .	4
2.1.3. <i>N</i> -Gram Language Model . . . . .	4
2.1.4. Estimating <i>N</i> -gram Probabilities . . . . .	5
2.2. Smoothing . . . . .	6
2.2.1. Good-Turing . . . . .	6
2.2.2. Discounting? . . . . .	8
2.2.3. Backing-Off? . . . . .	8
2.2.4. Kneser-Ney? . . . . .	8
2.3. Neural Networks . . . . .	8
<b>3. Table and Figure example</b>	<b>9</b>
<b>4. Summary and Conclusion</b>	<b>11</b>
<b>Bibliography</b>	<b>12</b>
<b>A. One or other Appendix Title</b>	<b>13</b>

# List of Figures

3.1. caption of the figure . . . . . 10

# List of Tables

2.1. Increase of parameters for n-gram size . . . . .	5
3.1. Table caption . . . . .	9

# Nomenclature

## Variables and functions

$p(x)$	Probability density function with respect to variable $x$ .
$P(A)$	Probability of event $A$ occurring.
$N(A)$	Number of times event $A$ occurred.
$E(x)$	Expectation of random variable $x$ .



**Acronyms and abbreviations**

LM	Language Model
MLE	Maximum Likelihood Estimation

# Chapter 1

## Introduction

...

# Chapter 2

## Literature Study

### 2.1. The Language Model

#### 2.1.1. The Need for the Statistical Language Model

Natural languages spoken by people are unlike programming languages, they are not made, but rather emerge naturally and tend to change over time. They are based on a set of grammatical rules, however spontaneous speech often deviates from it. Because of this, trying to program machines to interpret languages can be difficult. Rather than trying to model formal grammars and structures of a language, we find more success in basing them on statistical models. This is far less complex and still allows machines to interpret certain ambiguities, occurring often under natural circumstances.

One such ambiguity can be explained by an example. If a machine's purpose was to translate a user's speech to text, it could encounter the following problem; having to determine which of two words, that share acoustic characteristics, were said by the user. Such as "what do you see?" vs "what do you sea?". For the machine to interpret this correctly, we make use of a statistical language model. These models succeed because words do not appear in random order i.e. their neighbours provide much-needed context to them.

A statistical language model, in short, simply assigns a probability to a sequence of words. By doing so, the machine is able to choose the correct sequence of words based on their probabilities. To be clear, the language model does not assign a probability based on acoustic data, but rather the possible word sequences themselves.

### 2.1.2. Math behind the Model

The probability of a sequence of words is represented as follows:

$$P(\mathbf{w}(0, L - 1)) \quad (2.1)$$

with  $\mathbf{w}(0, L - 1) = w(0), w(1), \dots, w(L - 1)$  representing a sequence of  $L$  words. This probability is used by the machine in the speech-to-text example to successfully distinguish between the ambiguous words and choose the correct sequence. Much like a human uses the context of the given words.

The joint probability in 2.1 can be decomposed, using the definition of conditional probabilities, into a product of conditional probabilities:

$$P(\mathbf{w}(0, L - 1)) = \prod_{i=0}^{L-1} P(w(i) | \mathbf{w}(0, i - 1)) \quad (2.2)$$

Therefore the probability to observe the  $i^{th}$  word is dependent on the preceding  $i - 1$  words. To reduce complexity, we approximate 2.2 to only consider the preceding  $n - 1$  words:

$$\prod_{i=0}^{L-1} P(w(i) | \mathbf{w}(0, i - 1)) \approx \prod_{i=0}^{L-1} P(w(i) | \mathbf{w}(i - n + 1, i - 1)) \quad (2.3)$$

We have now derived the bases for the  $n$ -gram language model using the **Markov** assumption. Markov models are described as probabilistic models that predict some future unit without looking too far into the past [1].

### 2.1.3. N-Gram Language Model

In this report, we will only be focusing on one type of language model, the  $n$ -gram model. The  $n$ -gram model has several reasons for its success in becoming one of the preferred models. It is considered as being computationally efficient and simple to implement [2].

The next step would be to determine the size of  $n$ . This is also referred to as tuple size. Ideally, we would want a large tuple size, because words could still strongly influence others far down the sequence. However, the number of parameters estimated grows exponentially as tuple size increases. If we take a vocabulary size of 20,000 words, which is being modest, table 2.1 shows the number of parameters to be estimated for each tuple size [3].

**Table 2.1:** Increase of parameters for n-gram size

Tuple Size	Parameters
2 (bigram)	$20,000^2 = 4 \times 10^8$
3 (trigram)	$20,000^3 = 8 \times 10^{12}$
4 (four-gram)	$20,000^4 = 16 \times 10^{16}$
5 (five-gram)	$20,000^5 = 32 \times 10^{20}$

A five-gram model, although having more information available for better estimation, is not computationally practical. Four-gram models are considered to be right on the edge, but it is still largely left out and models tend to only incorporate bigrams and trigrams [3].

#### 2.1.4. Estimating N-gram Probabilities

The simplest way to estimate the probability is with a Maximum Likelihood Estimation, known as MLE. To obtain this estimate, we observe the  $n$ -gram counts from a training corpus. These counts are thereafter normalized to give [1]:

$$P(w_n|w_1...w_{n-1}) = \frac{N(w_1...w_n)}{\sum_k N(w_1...w_{n-1}, w_k)} \quad (2.4)$$

This equation can be further simplified into 2.5 as the count of a given prefix is equal to the sum of all  $n$ -gram counts containing that same prefix.

$$P(w_n|w_1...w_{n-1}) = \frac{N(w_1...w_n)}{N(w_1...w_{n-1})} \quad (2.5)$$

We now have a value that lies between 0 and 1, which we refer to as the relative frequency estimate.

A new problem arises because our training set and validation set have not necessarily seen the same  $n$ -grams. Even if we increase the size of the training set, the majority of words are still considered to be uncommon and longer  $n$ -grams containing these words are even rarer [3]. The validation set will most likely contain several unseen  $n$ -grams whose relative frequencies according to the  $n$ -gram model, built from the training set, are considered to be 0.

The probability of occurrence for a sequence of words should never be 0. Therefore some form of regularization is needed to prevent overfitting of the training data. This is where smoothing comes in. Essentially it decreases the probability of seen events and assigns

this leftover probability mass to unseen ones [3]. The next section will discuss different kinds of smoothing and how they are implemented.

## 2.2. Smoothing

### 2.2.1. Good-Turing

This smoothing technique was published by I. J. Good in 1953, who credits Alan Turing with the original idea [4]. It is based on the assumption that the frequency for each, in our case,  $n$ -gram follows a binomial distribution [1].  $N$ -grams occurring the same number of times are considered to have the same probability estimate.

We begin with the Good-Turing probability estimate:

$$P_{GT} = \frac{r^*}{N_\Sigma} \quad (2.6)$$

Where  $N_\Sigma$  is the total number of  $n$ -grams and  $r^*$  is a re-estimated frequency formulated as follows [3]:

$$r^* = (r + 1) \frac{E(C_{r+1})}{E(C_r)} \quad (2.7)$$

With  $C_r$  defining the count of  $n$ -grams that occur  $r$  times. Next we can approximate the expectations as counts and combine 2.6 and 2.7 to obtain:

$$q_r = \frac{(r + 1) \cdot C_{r+1}}{N_\Sigma \cdot C_r} \quad (2.8)$$

$q_r$  now denotes our Good-Turing estimate for the probability of occurrence of an  $n$ -gram occurring  $r$  times. For unseen  $n$ -grams,  $r = 0$ , giving us:

$$C_0 \cdot q_0 = \frac{C_1}{N_\Sigma} \quad (2.9)$$

where  $C_0 \cdot q_0$  is considered to be the probability of occurrence of any unseen  $n$ -gram.

From this we can now apply the Good-Turing technique to our language model, providing better accuracy. However, this technique has two concerns.

One is that our assumption in 2.8 is only viable for values of  $r < k$  (where  $k$  is a threshold, typically a value ranging from 5 to 10 [1, 3]). The MLE estimate of high frequency words is accurate enough that they do need smoothing.

The other concern being that it is quite possible for  $C_r$  to equal 0 for some value of  $r$ . This would result in a probability estimate of zero, leading us back to the start of our problem. To remedy this, one can smooth the values for  $C_r$ , replacing any zeroes before calculating the probability.

**2.2.2. Discounting?**

**2.2.3. Backing-Off?**

**2.2.4. Kneser-Ney?**

**2.3. Neural Networks**

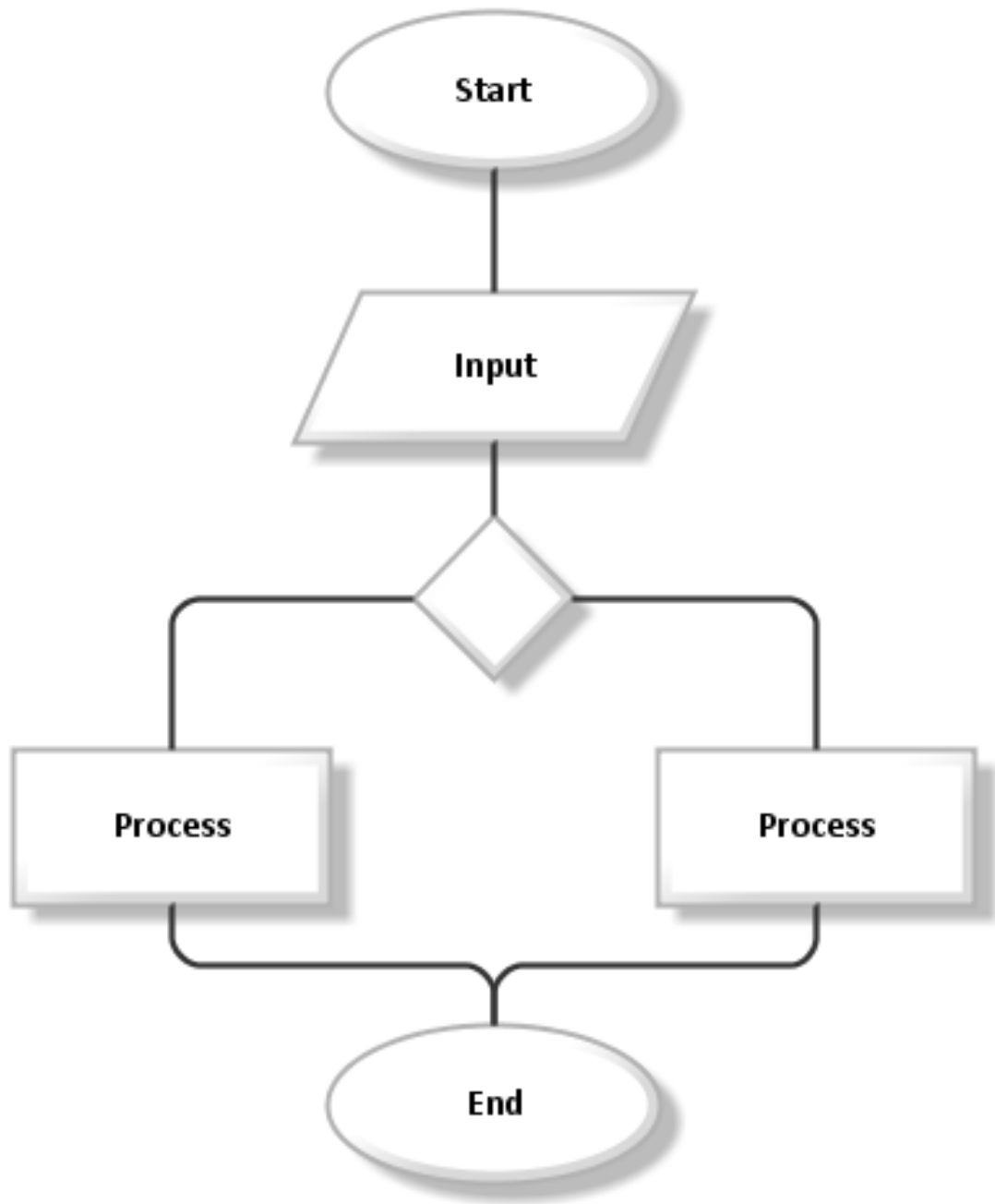


# Chapter 3

## Table and Figure example

**Table 3.1:** Table caption

Variable	1	2	3	4	5
X	35.4	23.5	21.5	21.2	22.9
Y	86.5	89.7	89.2	88.5	86.6
Z	70.6	72.2	71.8	70.9	69.4
Y	20	13	13	13	13



**Figure 3.1:** caption of the figure

This is a new page, showing how to refer to a figure like Figure 3.1.

## **Chapter 4**

### **Summary and Conclusion**

# Bibliography

- [1] D. Jurafsky and J. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, ser. Pearson international edition. Prentice Hall, 2009. [Online]. Available: <https://books.google.co.za/books?id=crxYPgAACAAJ>
- [2] T. R. Niesler, “Category-based statistical language models,” Ph.D. dissertation, University of Cambridge Cambridge, UK, 1997.
- [3] C. Manning, C. Manning, H. Schütze, and H. SCHUTZE, *Foundations of Statistical Natural Language Processing*, ser. Mit Press. MIT Press, 1999. [Online]. Available: <https://books.google.co.za/books?id=YiFDxbEX3SUC>
- [4] I. J. Good, “The Population Frequencies of Species and the Stimation of Population Parameters,” *Biometrika*, vol. 40, no. 3-4, pp. 237–264, 1953. [Online]. Available: <https://doi.org/10.1093/biomet/40.3-4.237>

# **Appendix A**

## **One or other Appendix Title**

This is some appendix.