# Explorations into Unsupervised Corpus Quality Assessment

Matje van de Camp
HAIT Master Thesis series nr. 08-004

Thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Arts in Communication and Information Sciences,
Master Track Human Aspects of Information Technology,
at the faculty of humanities
of Tilburg University

Thesis committee:

dr. M.W.C. Reynaert
dr. J.J. Paijmans

Tilburg University
Faculty of Humanities
Department of Communication and Information Sciences
Tilburg, The Netherlands
August 2008

**Abstract**

Corpora, large bodies of text, are of great importance to the field of Natural Language Processing. They are for instance used to train systems on specific tasks through machine learning. When a system is trained on a corpus of low quality, it will not provide reliable results. We search for a metric of corpus quality by comparing the vocabulary growth data, the Zipf slopes and the Pareto exponents of corpora of different sizes, compositions and qualities in Dutch and English. Vocabulary growth curves are a great tool for directly spotting major deviations in a text. Our method applies a linear regression on these curves, which unfortunately does not capture the small errors that contaminate a corpus, like mistakes in spelling. The Zipf slope and Pareto exponent do show explicable deviations when the quality of a corpus changes. However, their standard values of 1 and 2 respectively, which are reported in much of the literature, also deviate for different corpus sizes. Consequently, we did not find a solid metric of corpus quality, but a crude measure can certainly be derived from our results.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Ever since man started to write, there has been a struggle to find the best way to preserve these writings. From rock walls and clay tablets to paper, all materials used are subject to decay, resulting in a loss of information. Not just materials from for instance the Middle Ages are lost, but even newspapers printed only a few decades ago suffer from degradation and could be lost forever. Preserving them means saving our cultural heritage for future generations.
When it comes to cultural heritage, it is always preferable to preserve the original for full informational value. However, the technological advances of the last few decades have made it possible to save digital copies, aside from the fragile originals, which unfortunately have a limited lifespan. In digitized form, all books, newspapers and other writings can be preserved for decades, probably even centuries to come.
Besides the preservation of historical documents, digitization has a few other distinct advantages, depending on the chosen format. In the 1980s through 1990s many documents were transferred to microfilm. This is not only an easy way to store information, but it also saves the original form the information was in. Unfortunately, it can be very difficult and time-consuming searching through data on a microfilm, let alone further processing it.
Nowadays, it is more convenient to scan the documents, either the originals or the microfilms, into a computer. Storage is no problem, since entire libraries can be reduced to a single hard drive. Furthermore, the scanned images can be converted to a processable text format with the use of optical character recognition (OCR), which allows for a further processing of the text.
These days new texts are for the most part created directly in digital form, since more and more people use computers. Also, the explosive growth of the Internet and the number of e-mails sent everyday have resulted in a massive amount of data that is available for research. Web text mining is the

process of collecting large amounts of text from webpages and extracting useful information from it. Apart from the informational content of the texts collected, whether they are created recently or centuries ago, the texts themselves also provide great material for the investigation of human language and its structures.

## 1.1 Research motivation

Large collections of text, however they are accumulated, are of great interest to the relatively new field of computational linguistics or natural language processing (NLP). The main objective of NLP is:

> *to get computers to perform useful tasks involving human language, tasks like enabling human-machine communication, improving human-human communication, or simply doing useful processing of text or speech* [17].

Examples of the *useful processing of text* include:

- dialogue systems: computer systems that can recognize, understand and even respond to natural language;
- machine translation: systems that can automatically translate a text from one language to another;
- information extraction and retrieval;
- spelling and grammar correction.

A tool that is often used to train these systems is machine learning. Machine learning concerns itself with the development of algorithms used to instruct a computer about a certain process. In NLP, machine learning algorithms are used to train a computer system to recognize words and the structure of natural language so the system can carry out a specific task (semi-)automatically. The large bodies of text used as input for the systems are called *corpora*. A corpus can consist of unprocessed, plain text or text that has been preprocessed by another system or a human. An example of a preprocessed text is a text in which each word has been labeled with its appropriate syntactic category. In this case we speak of a *annotated* corpus. It is not unlikely that errors are made during the compilation or preprocessing of a corpus, whether it is done by a human or a computer. As we will show in this report, even the most state-of-the-art OCR systems still make

many mistakes. This results in a flawed corpus. For instance, Reynaert reports that over 21% of the types in a subset of the Reuters Corpus Volume 1 corpus are in fact typos [11]. The RCV1 is used by many linguistic researchers. The quality of a corpus used to train a system seriously affects the system's outcome. This means that when a system is trained on a corpus that is flawed to begin with, the results can be severely skewed and sometimes even unusable.
If there was a way to exactly determine, and maybe even resolve, the shortcomings of a corpus before using it to train a system, then NLP systems would produce far more reliable results. Here we report on our search for such a quality measure.

## 1.2 Research question

The goal of our research is to find a metric for the unsupervised assessment of corpus quality. With unsupervised we mean that we want to measure the quality without visually inspecting the contents of the corpus. Like [11], we assume that all the information we need to determine the quality of a sufficiently large corpus, is present in the corpus itself.
In order to find this metric, we will investigate several aspects of natural language which have been the subject of research before: power laws and vocabulary growth. Gelbukh and Sidorov [3] have already shown that the lexical richness of the language used in a corpus influences the power law distribution. We will focus our efforts on corpora in Dutch and English. English is a highly analytical language. Dutch uses a lot of compounds, which makes it a more inflective language.
Our main research question is: "Can power laws and vocabulary growth data be used to determine the quality of a corpus?". Other questions that will guide us in our research are: "How does a decrease in corpus quality affect word frequency distributions?", "Do corpora of lesser quality show a different vocabulary growth?" and "Does the estimation of vocabulary growth worsen for corpora of lesser quality?".

# Chapter 2

# Theoretical Background

## 2.1 Power laws: Zipf and Pareto

Many of the natural and man-made phenomena measured by scientists display a distribution that is centered around a typical value. A histogram of a distribution like this shows a distinct peak at the center value and is usually very narrow. However, there are phenomena that follow a more complicated distribution, for instance the distribution of word frequencies in human language.

It is known that in the production of human language only a few words get used many times, while many words only occur once or a few times. As a result the histogram of a word frequency distribution is highly right-skewed and has the shape of an "L". An interesting observation which was, most famously, made by George Kingsley Zipf is that a log-log plot of such a distribution displays a relatively straight line [20]. This shows us that we are dealing with a power law.

A power law follows a probability distribution function of the form:

$$p(x) = Cx^{-\alpha} \tag{2.1}$$

where C is a normalization constant and $\alpha$ is the exponent of the power law. The log-log transformation gives us the equation

$$log(p(x)) = log(C) - \alpha log(x) \tag{2.2}$$

which means that the log-log plot of the power law is a straight line with slope $-\alpha$ [5, 8].

Generally, there exist two kinds of power laws: discrete and continuous power laws. In a discrete distribution the measured variable can only take discrete values, usually positive integers. On the other hand, the measured values of a continuous power law follow a continuous distribution of real numbers [13]. *Zipf's law* for word frequencies is an example of a discrete power law. Zipf found that natural languages adhere to the following principle: the frequency of a word is inversely proportional to its rank in the frequency table. This means that the most common word, the word with rank 1, will occur approximately twice as much as the second ranked word, which in turn will occur approximately twice as much as the fourth ranked word, and so on. Zipf's law is formally defined as:

$$P_n \sim 1/n^a \tag{2.3}$$

where $P_n$ is the frequency of occurrence of the $n^{th}$ ranked item and $a$ is close to 1.[2] Zipf's law is not a law in the rigorous sense, but an empirical observation of the productivity of language. As with any power law, the easiest way to observe Zipf's law is to plot the log value of the word frequencies as a function of the log value of the word ranks. The resulting straight line has an exponent of approximately 1 [2, 4, 5, 8], which is referred to as the *Zipf slope* [15].
Another power law, which gives a continuous rather than a discrete probability distribution, is the *Pareto distribution.* It is named after Vilfredo Pareto, an Italian economist. He found this distribution when trying to describe the allocation of wealth among people. He observed that about 20% of the population is in control of 80% of the wealth. This 80-20 rule, also called the *Pareto principle*, has proven to hold true in many fields, including linguistics and the distribution of word frequencies.
In Zipf's law the word ranks appear on the x-axis, while the word frequencies appear on the y-axis. Pareto on the other hand, places the frequencies on the x-axis and their probabilities on the y-axis. This also affects the exponent of the power law, which lies around 2 in a Pareto distribution [2, 5]. Nevertheless, there exists a general relation between the rank of a word and the probability of its frequency. Equations 2.4 and 2.5 show how the Zipf slope, $\alpha$, and the Pareto exponent, $\beta$, can be calculated from one another:

$$\alpha = \frac{1}{\beta - 1} \tag{2.4}$$

$$\beta = \frac{1}{\alpha} + 1 \tag{2.5}$$

If $\alpha = 1$ then $\beta = 2$ [2].
A lot of research has been done to expose the existence of these power laws

[2]http://www.nist.gov/dads/HTML/zipfslaw.html

and their exponents in natural language. Ferrer i Cancho [7] gives a short overview of known situations where the Zipf slope significantly deviates from $\alpha \approx 1$. For instance, results have been reported of $\alpha \in [0.7, 0.9]$ for people in the early stages of schizophrenia. Advanced forms of schizophrenia however, lead to $\alpha = 1.5$. Military combat texts and young children's communication have also been reported as having $\alpha$¿1, with $\alpha = 1.4$ and $\alpha = 1.6$ respectively. In practice the plot of a Zipfian distribution seems to deviate from the straight line with exponent 1 for the highest ranks. This distortion in the tail has been considered a side-effect of the finite size of the texts used in research [2, 4]. However, Ferrer i Cancho and Solé [2] proposed a solution for this problem by dividing the text into two regimes, each having its own exponent. They tested this theory on the British National Corpus and found an exponent of 1.06 for the first 5,000-6,000 words, which constitute the first regime, or kernel lexicon. The second regime had an exponent of 1.97.

## 2.2 Vocabulary growth

Another aspect of human language and word frequencies which has received much attention is the growth rate of the vocabulary [6, 9, 14, 15, 16]. Each time a word gets added to a running text, this is either a word which has already appeared in the text or a new or unseen word. $N$ refers to the total number of words, called *tokens*, in the text. Each unique word is called a *type*. The total number of types in a text is denoted by $V$ and forms the total vocabulary of the text. Some words only appear once in a text. These are called *hapax legomena*, which is Greek for 'once said'. The hapax legomena, or simply *hapaxes*, are referred to by *V1*. Similarly, the words that are seen only twice or three times, the *dis legomena* and *tri legomena*, are referred to by *V2* and *V3* respectively.
As an illustration, consider the following nursery rhyme:

*Mary had a little lamb,*
*Little lamb, little lamb,*
*Mary had a little lamb,*
*Its fleece was white as snow.*

Not considering the punctuation marks, this text has 20 tokens, so $N$=20. When determining the number of types we have to make a choice whether to preserve the distinction between upper case and lower case. If we do, then "Little" and "little" are considered to be two separate types. This results in $V$=12. If we convert everything to lower case, we get $V$=11. Regardless of upper or lower case, six words appear only once in the text

(“Its”,“fleece”,“was”,“white”,“as” and “snow”), so *V1*=6. Using these numbers we can determine the type-token ratio (*V/N*) and the hapax-type ratio (*V1/V*), which are 0.6 and 0.5 respectively with *V*=12.
If we measure *N*, *V* and *V1* at regular intervals as a text progresses, we can see how the growth rates develop as the text goes on. Figure 2.1 displays the

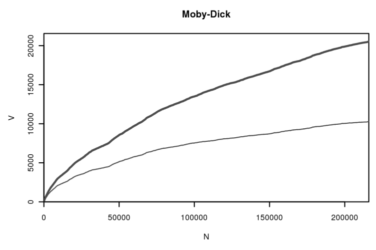

**Figure 2.1:** Vocabulary growth curve for Herman Melville's *Moby-Dick*. The thick, upper line shows the number of types (*V*); the thin, lower line shows the number of hapaxes (*V1*). The type-token ratio is 20531/215994 = 0.1; the hapax-type ratio is 10282/20531 = 0.5.

growth for the total text of Herman Melville's *Moby-Dick*. In the first chapters of the book, the number of types and hapaxes grows at a fast rate. The growth slows down substantially as the text progresses. This is explained by the fact that in the beginning of a coherent story many new concepts and characters are introduced, which get called upon again later in the story. At the end of the text, when all tokens are considered, we see that the growth still has not stopped. We can conclude from this that if the story was to go on, new words would still be added to the vocabulary, although at a much slower pace than at the beginning.
Also, the hapax-type ratio is approximately 0.75 at the beginning of the text, but decreases to roughly 0.5 after only 800 tokens. An even more drastic decrease is seen for the type-token ratio. It starts at around 0.7 and rapidly decreases to 0.1 after only 700 tokens.
Since vocabulary growth curves show the growth for the entire text, they can be used to make estimates about the growth of smaller or larger texts. A method of estimating the growth for sizes up to the entire text is called *interpolation*. A method of estimating the growth beyond the size of the text is called *extrapolation*.

### 2.2.1 Interpolation

In mathematics, curve fitting is used to find a function which best fits a series of observed data points. Interpolation is a special case of curve fitting

in which a function is created that goes exactly through the observed data points.
Evert and Baroni [14] have developed the zipfR package for the R statistical environment. It contains functions to calculate the interpolated growth, as well as the actual, or observed, growth. The interpolation function takes a sample of the data points in the observed data and fits a function on this sample. It is based on the binomial interpolation formula described by Baayen [16]:

$$E[V_{N_0}(m, N)] = \sum_{k \geq m} V(k, N_0)\binom{k}{m}(\frac{N}{N_0})^m(1 - \frac{N}{N_0})^{k-m} \tag{2.6}$$

$$E[V_{N_0}(N)] = V(N_0) + \sum_{m=1}^{N_0}(-1)^{m-1}V(m, N_0)(\frac{N}{N_0} - 1)^m \tag{2.7}$$

Equations 2.6 and 2.7 respectively give the conditional spectrum elements for a sample size of $N$ given the frequency spectrum of $N_0$ tokens and the conditional vocabulary size. Comparison of the observed and interpolated curves can show unexpected variations in the observed data and expose possible errors in the text.
As an illustration, consider figure 2.2. Here we have plotted the observed and interpolated growths for a Dutch translation of *Moby-Dick*. The book was scanned and OCRed without looking at its contents. Instead of forming a smooth curve, the observed growth of the number of types is horizontal between roughly 170,000 and 180,000 tokens. This means that there are no new types added to the vocabulary in this section. Also, the hapax curve goes down in this part of the text. This implies that some of the tokens that were seen only once in the text preceding this section are repeated here, while no new hapaxes are added.
However, the interpolated growth expects both *V* and *V1* to continue to grow for the full length of the corpus. This deviation leads us to suspect that the observed text contains a major flaw. Visual inspection of the book confirms this suspicion as it shows that an entire section of the text was duplicated.

### 2.2.2 Extrapolation

Extrapolation is also a form of curve fitting and is very similar to interpolation. The main difference is that extrapolation is used to predict data points *outside* the range of known data points. Unfortunately, prediction through extrapolation is subject to great uncertainty.

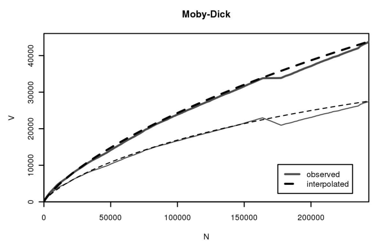

**Figure 2.2:** Vocabulary growth curve for a scanned and OCRed copy of S. Westerdijk's Dutch translation of Herman Melville's *Moby-Dick*. The thick, upper lines show the number of types (*V*); the thin, lower lines show the number of hapaxes (*V1*). Between approximately 170,000 and 180,000 tokens the observed growth clearly deviates from the interpolated or expected growth.

The estimations needed for the extrapolation are computed using Large-Number-of-Rare-Events (LNRE) models [16]. Evert and Baroni [6] researched the extrapolation quality using several models. They found that extrapolation models in general give plausible results when predicting the vocabulary size *V* up to 2 times the corpus size. For *V1*, the number of hapaxes, the accuracy of extrapolation is much lower. The finite Zipf-Mandelbrot model (fZM) and the Generalized Inverse Gauss Poisson (GIGP) model gave the best overall results. Both the fZM and the GIGP models are incorporated in the zipfR package.

# Chapter 3

# Methods

In order to see if we can determine the quality of a corpus we need some example corpora of which the exact quality is known. Since quality assessment is the main objective of our study, this may seem to lead us into a vicious circle. Fortunately, there are many high quality texts and corpora available that have been used extensively in linguistic research. By systematically introducing errors into these corpora we can control not only how much the quality deteriorates, but also in what way.

## 3.1 Corpora

The books used were all downloaded from the website of Project Gutenberg.[1] Since the texts are proofread before being placed on the website, we can safely assume they are of a high quality.

### 3.1.1 Alice's Adventures in Wonderland

The book *Alice's Adventures in Wonderland* by Lewis Carroll is a relatively short novel of a surrealistic genre. We have downloaded the entire English text from Project Gutenberg. The text contains about 30,000 tokens.
There are several reasons why we chose this particular book. First of all, it has been used by many researchers before us, especially Baayen [15, 16] and also Powers [1]. So there is a lot of data available for comparison.
Secondly, previous research has shown that the surrealistic nature of the story has an influence on the vocabulary growth and the power law exponents. The book was compiled from different children's stories, which were

[1] www.gutenberg.org

originally delivered verbally. The author introduces new characters and situations throughout the book every time a new adventure begins. Normally this is done mainly at the beginning, after which they get called upon again and again in the rest of the text. In *Alice*, many characters disappear just as quickly as they entered and each chapter has its own vocabulary [1]. As a result the text is more fragmented and incoherent.

### 3.1.2 Anna Karenina

The events described in Leo Tolstoy's novel *Anna Karenina* are of a realistic nature. Also, there are only about a dozen characters featured in the story, which puts it in great contrast to the amount of wacky characters that appear in *Alice*, considering its size.
For our research we used a Dutch translation of the book we found on Project Gutenberg. It consists of nearly 220,000 words.

### 3.1.3 Moby-Dick

Like *Alice*, Herman Melville's *Moby-Dick* has also been a popular subject in linguistic research. The English text consists of a little more than 210,000 words. It is a fictional story which includes large sections devoted to explaining the whaling business of the 19th century. Both with respect to topicality and events the book has a coherent structure.

### 3.1.4 British National Corpus

The British National Corpus (BNC) is a collection of about 100 million words of modern British English. All texts were created in the second half of the 20th century and cover a wide array of topics. Written texts like books, papers and articles make up 90% of the corpus, while the remaining 10% consists of transcripts of spoken text recorded in a variety of situations. The size of the individual texts is also subject to great variation and ranges from as little as 28 to over 465,000 words.
Even though the corpus is fragmented in this way, the texts are grouped together according to subject or theme, creating the illusion of a coherent story with minimal shifts in topic. Nevertheless, since there are many different authors (and speakers), it still is far less homogeneous than *Anna Karenina* or *Moby-Dick*.
For our research we used the first version of the BNC, which was published in 1994. When we received the corpus, it had already been tokenized. We also received a normalized version of the corpus, in which all punctuation

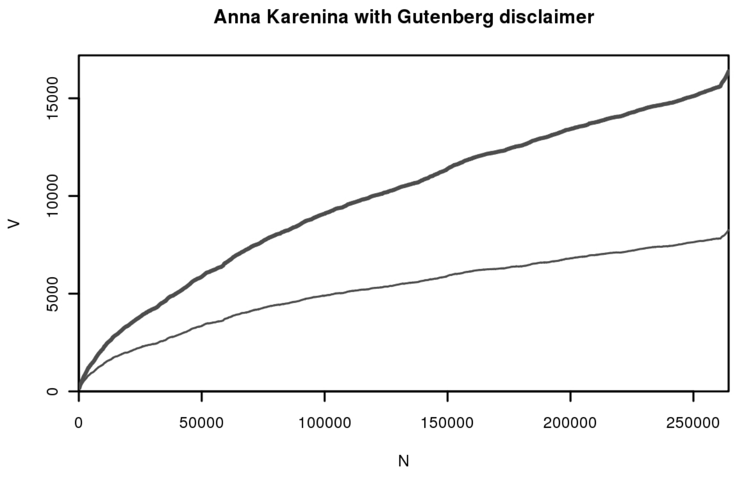

**Figure 3.1:** Vocabulary growth curve for *Anna Karenina* including the Gutenberg disclaimer. The thick, upper line shows the number of types ($V$); the thin, lower line shows the number of hapaxes ($V1$). The type-token ratio is 0.06; the hapax-type ratio is 0.5.

marks have been replaced by a dot (.) and all numbers have been converted to 3. Normalizing a corpus in this manner makes it more coherent, since the appearance of specific punctuation marks and numbers in a text is very arbitrary and can result in an high number of hapaxes.

## 3.2 Preprocessing

The corpora have been processed in several ways to eliminate factors that form a possible threat to the corpus quality. To determine how great the effect of each 'shortcoming' is, we use both the unprocessed and the processed versions of the corpora in our experiments.
The preprocessing techniques have not been applied to all corpora. For instance, our version of the BNC had already been tokenized and normalized, as discussed in section 3.1. Since this corpus is compiled from many relatively short texts of different origin and genre, we decided not to subject it to any further processing, but rather compare the results for the corpus as a whole and the separate documents in the corpus.

### 3.2.1 Gutenberg disclaimer

The corpora gathered from Project Gutenberg all contain a disclaimer in English. This may not seem as a serious problem for *Alice* and *Moby-Dick*,

since they are also written in English. However, the information given in the disclaimer is of an entirely different nature than the books themselves, and thus influences the coherence of the text.
On top of that, our version of *Anna Karenina* is in Dutch. The disclaimer consists of 3,000 words, which is over 1% of the corpus. A large section in a different language seriously compromises the homogeneity of a text. Even more so, it has a major influence on the vocabulary growth. Figure 3.1 shows a sudden burst of new types and hapaxes at the end of the corpus where the disclaimer is located. A deviation like this makes it unnecessarily difficult to fit a function to the data with interpolation and makes it next to impossible to make predictions through extrapolation.

### 3.2.2 Tokenization

Tokenization is the process of separating words from punctuation marks. Since the punctuation marks are not removed from the text, they are counted as tokens themselves, increasing the value of $N$. On the other hand, in a non-tokenized text, words that have punctuation marks attached to them would be counted as new words in the vocabulary. Also, punctuation marks are used in abundance, so counting them as tokens will not seriously affect the number of hapaxes in the text.

### 3.2.3 Lower case

As discussed in section 2.2, transforming everything in a corpus to lower case is a way of making sure all occurrences of a word get counted as the same type, no matter if they are the first word in a sentence and therefore capitalized. A minor side effect is that for instance names that are spelled the same as an existing word do not count as a separate type. But this is very uncommon and the effect is probably negligible.

### 3.2.4 Error insertion

There are multiple ways in which a corpus can get contaminated with errors. Also, the possible types of errors that can occur are numerous. Text mined from web pages contains a lot of tags and meta data. Transcriptions of audio, whether they are produced by an automatic or a human transcriber, can contain errors due to unintelligible speech. Even the most state-of-the-art OCR software is not 100% accurate. It will misrecognize at least a small percentage of the characters or words, thus contaminating the corpus.
Instead of using an already flawed corpus, we inserted errors into our own

high quality corpora. This gives us more control over what happens and leaves us with a gold standard version and a flawed version, which we can then compare to each other.
For *Moby-Dick* and *Alice*, we used a Perl program, `ramdomreplace.10.pl`, to insert a certain amount of spelling errors. The spelling errors are taken from a list of correctly spelled English words with their possible misspellings compiled by Reynaert [18]. The program requires as an argument a percentage of errors to insert. For both corpora we used 25% and 50% error insertion. Of course, not all words on the list will be present in each corpus, so the percentage entered will not be the exact measure of quality loss.
To measure the amount of quality loss we compared the gold standards with the 25% and 50% versions using the `wordacc` program in the OCR Frontiers Toolkit [19], which measures word accuracy between two corpora. For *Moby-Dick* 25% error insertion resulted in a quality loss of 9%; 50% error insertion decreased the quality with 16%. For *Alice* the resulting quality decrease was 8% and 15%, respectively.
With *Anna Karenina* we chose a different approach. The text was first printed out and then scanned and OCRed using three different OCR software packages: ABBYY Finereader 8.0 Professional Edition, ABBYY Finereader 9.0 Professional Edition and ScanSoft OmniPage Professional 16. Since these programs are all state-of-the-art, the quality loss was minimal. After measuring the accuracy with the OCR Frontiers Toolkit, ABBYY 9.0 turned out to have performed the worst, with a quality loss of 0.21% on character level and 0.43% on word level. This is surprising, since it is supposed to be an improved version of ABBYY Finereader 8.0. Both other OCR versions were not used in the experiments.

## 3.3 Experiments

### 3.3.1 Alice, Anna Karenina and Moby-Dick

#### Vocabulary growth

For each corpus we plotted the observed, interpolated and extrapolated vocabulary growths. The observed growth was measured at regular intervals of 500 tokens. Both the interpolated and extrapolated growths were computed by taking a random sample of the frequency spectra of each text.
Normally interpolation is done for values smaller than $N$ and extrapolation for values larger than $N$. We decided to calculate the estimates over the entire text size in order to see if either method would give an accurate prediction of the actual growth. Two different models were used in the extrapolation: fi-

nite Zipf-Mandelbrot and Generalized Inverse Gauss Poisson. As mentioned in section 2.2.2, Evert and Baroni found these two models to give the best results [6].
We fitted a linear regression to each plot. From the slope of the regression we calculated the exponent by taking $10^{slope}$.
Since *Alice* is almost 10 times smaller than both *Anna Karenina* and *Moby-Dick*, it would be hard to tell if and in what way the corpus size has an effect on the results. Baayen suggests two solutions for this problem ([15], pages 245-248). One solution is to reduce all corpora to the same size. Another option is to develop better statistical models. Keeping practicality in mind, we chose the former and have repeated the experiments with the larger corpora reduced to the size of the smaller one.

#### Zipf and Pareto

The Zipf slope for each text was computed by fitting a linear regression to the Zipfian distribution. We did not divide the distribution into two regimes as described in [2]. In much of the literature a Zipf slope of 1 is reported without distinguishing different regimes. We want to measure if and how the Zipf slope changes when corpus quality deteriorates. Fitting two separate functions to the same data might mask the effect of poor corpus quality on the results.
The Pareto exponent was calculated using the software provided by Aaron Clauset as a companion to [13].[2] He describes a few different methods of fitting the power law to the data, which are also incorporated in the software. One way of doing it is by performing a least-squares linear regression to the log-log histogram of the data. According to Clauset, this method is far from reliable and one should use the method of maximum likelihood instead. We used both methods on our data.
The fitting function allows the threshold at which the power law is suspected to start to be defined. For instance, it is possible that the first few data points do not follow the same probability distribution as the rest of the data. This would be shown in the log-log plot as a deviation from the straight line for low values of $x$. By setting the threshold at the lowest value of $x$ on the straight line, the data below the threshold is not taken into account when fitting the power law and the resulting fit is supposedly better.
In calculating the Zipf slope we did not leave out any of the data points when fitting the power law. Therefore we chose not to do this for the fitting of the Pareto exponent either and set the threshold to 1.

[2]http://www.santafe.edu/∼aaronc/powerlaws/powerlaws_full_v0.0.6-2008-04-25.tgz

To automate the estimation of the vocabulary growth and the fitting of the power laws we have developed a small shell script and an accompanying R program. The script is called power_exploRe.sh and can be found in section B.1, together with the R program.

### 3.3.2 British National Corpus

The same experiments mentioned in section 3.3.1 were performed on both the tokenized version of the BNC and the normalized version. Since the BNC is a very fragmented corpus with multiple authors, this can have a serious effect on the outcome. In order to visualize the effect of this incoherence, we repeated the experiments on the tokenized version divided into the original documents that make up the corpus.[3]

By averaging the results from the separate documents, the differences in style, topic or genre get normalized and we create the illusion of a single author. This makes it easier to compare the results from the BNC with the results of the Gutenberg books.

[3]http://www.natcorp.ox.ac.uk/docs/userManual/bncIndex.html

# Chapter 4

# Results

## 4.1 Tokenization and lower case

### 4.1.1 Anna Karenina

Table 4.1 shows the Zipf slopes and the Pareto exponents for the unprocessed and preprocessed versions of *Anna Karenina*, our only Dutch corpus. The top half of the table displays results from the corpus including the Project Gutenberg disclaimer, the bottom half without it.
The Zipf slopes are close to 1. The Pareto exponents that were fitted using linear regression are all close to 2. These values are in line with expectations. They also show that when the Zipf slope increases, the Pareto exponent decreases. This is easily explained by equations 2.4 and 2.5.
The Pareto exponents that were acquired using maximum likelihood estimation show more diversity and come closer to 3. When applying equation 2.5 to these results they predict a Zipf slope of around 1.5, which is far from the observed results. Also, the values do not automatically decrease with higher Zipf slopes.
Both tokenization and lower case transformation seem to lead to a higher Zipf slope and a lower Pareto LR exponent, although the observed effect is greater with tokenization. The effect accumulates when tokenization and lower case are both applied to the corpus.
However, for the Pareto MLE exponent, lower case seems to have the opposite effect, as the exponent actually increases, and the effect of tokenization is much more evident. The exponent decreases over 7% with respect to the raw version of the corpus.
Contrary to what we expected, the results without the disclaimer do not differ substantially from the results including the disclaimer. For the Pareto MLE exponents they are practically the same.

**Anna Karenina**

| | | *Zipf slope* | *Pareto MLE* | *Pareto LR* |
|---|---|---|---|---|
| **Disclaimer** | *raw* | 1.02 | 2.97 | 1.97 |
| | *tokenized* | 1.05 | 2.75 | 1.95 |
| | *lower case* | 1.04 | 2.98 | 1.96 |
| | *tokenized & lower case* | 1.06 | 2.87 | 1.93 |
| **No disclaimer** | *raw* | 1.03 | 2.97 | 1.96 |
| | *tokenized* | 1.06 | 2.75 | 1.94 |
| | *lower case* | 1.04 | 2.98 | 1.95 |
| | *tokenized & lower case* | 1.08 | 2.89 | 1.92 |

**Table 4.1:** Zipf slopes and Pareto exponents for the unprocessed and preprocessed versions of *Anna Karenina*. Pareto MLE refers to a Pareto fitting with maximum likelihood estimation; Pareto LR refers to a Pareto fitting with the least-squares linear regression method. The threshold was set to 1 in both cases. The number of tokens in the raw versions are $N$ = 226,446 and $N$ = 223,340 respectively.

**Anna Karenina**

| | *observed* | *interpolated* | *extrapolated* | *extrapolated* |
|---|---|---|---|---|
| | | | *fZM* | *GIGP* |
| *raw* | 1.34 (4302) | 1.34 (4723) | 1.34 (4637) | 1.34 (4673) |
| *tokenized* | 1.12 (3188) | 1.13 (3713) | 1.13 (3605) | 1.13 (3633) |
| *lower case* | 1.33 (4148) | 1.33 (4577) | 1.33 (4493) | 1.33 (4527) |
| *tokenized & lower case* | 1.14 (3108) | 1.14 (3697) | 1.15 (3576) | 1.14 (3604) |

**Table 4.2:** Vocabulary growth exponents for the unprocessed and preprocessed versions of *Anna Karenina*, with the Project Gutenberg disclaimer. The exponents were obtained by performing a linear regression on the growth curves. The intercepts of the regression lines are between parentheses. Raw: $N$ = 226,446; $V$ = 31,885. Tokenized: $N$ = 264,388; $V$ = 16,374. Lower case: $N$ = 226,446; $V$ = 30,958. Tokenized and lower case: $N$ = 258,360; $V$ = 17,554.

### 4.1.2 Moby-Dick and Alice

The results for the Zipf slopes and Pareto exponents of *Moby-Dick* and *Alice* are listed in tables 4.3 and 4.5. The differences between the results of the corpora with the Gutenberg disclaimer or without it were minimal, so only the results with the disclaimer are reported here.
The results for *Moby-Dick* (table 4.3) are in line with the results reported for *Anna Karenina.* The Zipf slopes and Pareto LR exponents are close to 1 and 2, as expected. The raw version of the corpus actually gives 'perfect'

results with a Zipf slope of exactly 1.00 and a Pareto LR exponent of 2.00. The effect of tokenization on both figures is slightly greater than the effect of the lower case transformation, which is only shown in the Pareto LR exponent. When applied together, the observed effect is approximately twice as big.
The Pareto MLE exponents are a bit lower than those for *Anna Karenina*. Here they do display a decrease with higher Zipf slopes and also the effects of tokenization and lowercase follow the same trend as they do in the Pareto LR exponents. The combination of tokenization and lower case, however, shows a smaller decrease than tokenization alone. In this case lower case seems to diminish the effect of tokenization to a certain degree.
The Zipf slopes and Pareto LR exponents for *Alice* (table 4.5) deviate from the expected values of 1 and 2. The values of the Zipf slopes converge to 0.9. The difference in the Pareto LR exponents follows what would be expected from equation 2.4 and center around 2.1.
Interestingly enough, the effect of tokenization on the Zipf slopes and Pareto LR exponents is turned around. Instead of increasing the Zipf slope and decreasing the Pareto LR, the Zipf slope decreases and the Pareto LR consequently increases. Although the effect of lower case is still visible as it was in *Moby-Dick* and *Anna Karenina*, it does not make any difference when it is applied in conjunction with tokenization.
For *Alice* the deviations in the Pareto MLE exponents are the greatest. The values do however follow the same trend that we saw in *Moby-Dick*. The only difference is that in the case of tokenization ánd lower case the decrease is much bigger. Here the effects of tokenization and lower case are accumulated like they are in the Zipf and Pareto LR results.

## 4.2 Vocabulary growth

In the same way that we determined the slope of the Zipf and Pareto distributions by fitting a linear regression to it, we can fit a straight line to the vocabulary growth data. Although the growth of a text follows a curve rather than a straight line, the fitted regression will give us an idea of the overall progress of the growth.
Table 4.2 gives the results for the Gutenberg version of *Anna Karenina*. The exponents of the regression lines hardly differ between the observed, interpolated and extrapolated growths. However, the intercepts of the line, which are the numbers between parentheses, show that the lines do not cut through the $x$-axis at the same point and therefore are not at the same height in the plot. This means that the interpolated and extrapolated growths expect

**Moby-Dick**

| | *Zipf slope* | *Pareto MLE* | *Pareto LR* |
|---|---|---|---|
| *raw* | 1.00 | 2.88 | 2.00 |
| *tokenized* | 1.02 | 2.76 | 1.97 |
| *lower case* | 1.02 | 2.83 | 1.98 |
| *Tokenized & lower case* | 1.04 | 2.79 | 1.96 |

**Table 4.3:** Zipf slopes and Pareto exponents for the unprocessed and preprocessed versions of *Moby-Dick*. The number of tokens in the raw version is $N$ = 217,852.

**Moby-Dick**

| | *observed* | *interpolated* | *extrapolated fZM* | *extrapolated GIGP* |
|---|---|---|---|---|
| *raw* | 1.46 (4531) | 1.46 (5105) | 1.46 (4851) | 1.46 (4894) |
| *tokenized* | 1.19 (3956) | 1.19 (4482) | 1.19 (4252) | 1.19 (4300) |
| *lower case* | 1.44 (4332) | 1.43 (4976) | 1.44 (4714) | 1.43 (4758) |
| *tokenized & lower case* | 1.19 (3763) | 1.19 (4321) | 1.19 (4096) | 1.19 (4140) |

**Table 4.4:** Vocabulary growth exponents for the unprocessed and preprocessed versions of *Moby-Dick*. The exponents were obtained by performing a linear regression on the growth curves. The intercepts of the regression lines are between parentheses. Raw: $N$ = 217,852; $V$ = 38,638. Tokenized: $N$ = 252,372; $V$ = 21,970. Lower case: $N$ = 217,852; $V$ = 36,835. Tokenized and lower case: $N$ = 249,805; $V$ = 21,349.

**Alice**

| | *Zipf slope* | *Pareto MLE* | *Pareto LR* |
|---|---|---|---|
| *raw* | 0.91 | 3.05 | 2.09 |
| *tokenized* | 0.88 | 2.91 | 2.13 |
| *lower case* | 0.93 | 2.99 | 2.07 |
| *tokenized & lower case* | 0.88 | 2.81 | 2.13 |

**Table 4.5:** Zipf slopes and Pareto exponents for the unprocessed and preprocessed versions of *Alice*. The number of tokens in the raw version is $N$ = 30,404.

**Alice**

| | *observed* | *interpolated* | *extrapolated fZM* | *extrapolated GIGP* |
|---|---|---|---|---|
| *raw* | 1.57 (760) | 1.59 (945) | 1.60 (921) | 1.59 (933) |
| *tokenized* | 1.20 (708) | 1.22 (870) | 1.22 (827) | 1.22 (838) |
| *lower case* | 1.53 (728) | 1.54 (915) | 1.55 (895) | 1.54 (906) |
| *tokenized & lower case* | 1.19 (657) | 1.21 (809) | 1.21 (768) | 1.21 (777) |

**Table 4.6:** Vocabulary growth exponents for the unprocessed and preprocessed versions of *Alice*. The exponents were obtained by performing a linear regression on the growth curves. The intercepts of the regression lines are between parentheses. Raw: $N$ = 30,404; $V$ = 6,754. Tokenized: $N$ = 36,829; $V$ = 3,780. Lower case: $N$ = 30,404; $V$ = 6,310. Tokenized and lower case: $N$ = 36,397; $V$ = 3,533.

more types throughout the text than there in fact are.
On a side note, it may seem odd that the intercept of the lines is not 0, since obviously at the beginning of a text, when $x = 0$, we have not seen any words and $V = 0$. This is caused by the rapid growth at the beginning of the text and the fact that the regression tries to find a fit that is representative for most of the data points and not just the first few.
Tables 4.4 and 4.6 show the same results for *Moby-Dick* and *Alice*. With *Alice* there is a slight difference between the observed and expected growths, but this difference in slope does not explain the large differences between the intercepts.
For both *Anna Karenina* and *Moby-Dick* an fZM extrapolation of the raw text seems to give the best fit, since the regression comes closest to the observed growth. The difference between the observed and fZM intercepts is 7.8% for *Karenina* and 7.1% for *Moby-Dick*. This may seem like a lot, but viewed in perspective of the total number of types in the text, this translates to respectively 1.1% and 0.8%.
Interpolation overall, but especially on the tokenized and lower case version, gives the worst results for both corpora. The differences with the observed intercepts are 19% and 14.8% respectively, which translates to 3.4% and 2.6% in light of the whole text.
For the vocabulary growth of *Alice* fZM gives the best fit too, only here tokenization has a strong positive effect on the extrapolation. Lower case seems to diminish this effect slightly, so applying only tokenization gives the best result. The difference between the observed and fZM intercepts for the tokenized corpus is 16.9%, which translates to 3.2% overall. A possible explanation for this effect could be the fact that tokenization increases the number of tokens. *Alice* is a relatively small corpus and it is extra difficult to determine expected values when the number of observed values is minimal. Increasing the number of tokens also increases the number of available measurements to sample from and may therefore lead to a more accurate estimation.
Again, the interpolated results are the worst. But this time it is the lower case version of the corpus that differs the most from the observed growth. This too can be attributed to the small size of *Alice*, since lower casing decreases the number of types even further and thus inflates any difference between the observed and expected value of $V$.
The most striking difference between *Karenina* and *Moby-Dick* on the one hand and *Alice* on the other, is that tokenization has a negative effect on the expected growths of the first and a positive effect on the expected growth of the latter.

## 4.3 Corpus size

To determine how much of the differences between *Alice* and the other two corpora can be attributed to corpus size, we conducted the same experiments on versions of *Karenina* and *Moby-Dick* which are reduced to the same number of tokens ($N$) as *Alice*.
When looking at the Zipf and Pareto results, we see that for both corpora the Zipf slopes go up, with regards to the full version of the text, and the Pareto LR exponents go down. The values do come closer to those of *Alice*, especially in the case of the raw versions and, to a slightly lesser degree, the lower case versions.
However, both tokenization and the combination of tokenization and lower case have a positive effect for *Karenina* and *Moby-Dick*, in the sense that they bring the Zipf slope closer to 1 and the Pareto LR exponent closer to 2. With *Alice* tokenization had a negative effect.
Considering the Pareto MLE exponents, all three of the small corpora show the same trend. The exponent decreases when tokenization is applied. Although, overall the exponents of *Karenina* and *Moby-Dick* are much higher than those of *Alice*.
Again the exponents of the vocabulary growth fits do not differ that much, if at all, between observed and expected. It is interesting to see that interpolation and extrapolation both expect less types than there are observed in the short version of *Karenina*: the intercepts decrease for the expected growths. This is not seen in the results for *Moby-Dick*, where the intercepts increase like they do with *Alice* and the full version of the corpus.
For *Karenina* the interpolation of the lower case text now gives the best estimate, with a difference in intercept of only 0.2%, translated to a mere 0.02% in light of the total number of types. The fZM extrapolation still gives the best fit in case of *Moby-Dick*, with a difference of 12.2% and 1.2% respectively for the raw text.
Where tokenization seemed to have a positive effect on the prediction of vocabulary growth in *Alice*, this is not the case in either *Karenina* or *Moby-Dick*. Lower case conversion however, does improve estimations for *Karenina*. This effect was also visible for the full version and thus can not be attributed to corpus size.
To sum it up, reducing corpus size has an effect on the Zipf slope, which decreases to values below 1, and consequently on the Pareto LR exponent, which increases to values above 2. The Pareto MLE exponent also increases. Vocabulary growth estimation is a more complex matter, which seems not only to be subject to corpus size, but perhaps also language or genre.

**Anna Karenina *N*=Alice**

| | *Zipf slope* | *Pareto MLE* | *Pareto LR* |
|---|---|---|---|
| *raw* | 0.92 | 3.36 | 2.06 |
| *tokenized* | 0.94 | 2.97 | 2.05 |
| *lower case* | 0.94 | 3.38 | 2.04 |
| *tokenized & lower case* | 0.95 | 3.08 | 2.03 |

**Table 4.7:** Zipf slopes and Pareto exponents for the unprocessed and preprocessed versions of *Anna Karenina*, reduced to the size of *Alice*. The number of tokens in the raw version is $N$ = 30,404.

**Anna Karenina *N*=Alice**

| | *observed* | *interpolated* | *extrapolated* | *extrapolated* |
|---|---|---|---|---|
| | | | *fZM* | *GIGP* |
| *raw* | 1.60 (917) | 1.60 (910) | 1.61 (881) | 1.60 (890) |
| *tokenized* | 1.29 (945) | 1.30 (897) | 1.31 (862) | 1.30 (871) |
| *lower case* | 1.58 (882) | 1.58 (881) | 1.58 (850) | 1.58 (858) |
| *tokenized & lower case* | 1.31 (887) | 1.32 (861) | 1.32 (831) | 1.32 (838) |

**Table 4.8:** Vocabulary growth exponents for the unprocessed and preprocessed versions of *Anna Karenina*, reduced to the size of *Alice*. The exponents were obtained by performing a linear regression on the growth curves. The intercepts of the regression lines are between parentheses. Raw: $N$ = 30,404; $V$ = 6,787. Tokenized: $N$ = 36,829; $V$ = 4,824. Lower case: $N$ = 30,404; $V$ = 6,573. Tokenized and lower case: $N$ = 36,397; $V$ = 4,932.

**Moby-Dick *N*=Alice**

| | *Zipf slope* | *Pareto MLE* | *Pareto LR* |
|---|---|---|---|
| *raw* | 0.92 | 3.61 | 2.08 |
| *tokenized* | 0.93 | 3.24 | 2.06 |
| *lower case* | 0.95 | 3.57 | 2.05 |
| *tokenized & lower case* | 0.95 | 3.23 | 2.05 |

**Table 4.9:** Zipf slopes and Pareto exponents for the unprocessed and preprocessed versions of *Moby-Dick*, reduced to the size of *Alice*. The number of tokens in the raw version is $N$ = 30,404.

**Moby-Dick *N*=Alice**

| | *observed* | *interpolated* | *extrapolated* | *extrapolated* |
|---|---|---|---|---|
| | | | *fZM* | *GIGP* |
| *raw* | 1.80 (793) | 1.80 (894) | 1.80 (889) | 1.80 (898) |
| *tokenized* | 1.46 (857) | 1.45 (978) | 1.45 (973) | 1.45 (986) |
| *lower case* | 1.75 (709) | 1.75 (866) | 1.75 (866) | 1.75 (874) |
| *tokenized & lower case* | 1.46 (755) | 1.44 (932) | 1.44 (934) | 1.44 (945) |

**Table 4.10:** Vocabulary growth exponents for the unprocessed and preprocessed versions of *Moby-Dick*, reduced to the size of *Alice*. The exponents were obtained by performing a linear regression on the growth curves. The intercepts of the regression lines are between parentheses. Raw: $N$ = 30,404; $V$ = 8,272. Tokenized: $N$ = 36,829; $V$ = 6,564. Lower case: $N$ = 30,404; $V$ = 7,889. Tokenized and lower case: $N$ = 36,397; $V$ = 6,343.

## 4.4 Flawed corpora

For the flawed versions of our corpora tokenization and lower case showed the same effects as they did for the gold standards. Therefore we will only report the results obtained from the raw versions.
As described in section 3.2.4 the “low” quality version of *Anna Karenina* was created by performing OCR on the gold standard. The resulting corpus still has an accuracy of 99.6%. Table 4.11 shows that such a minor decrease in quality hardly has an effect on the Zipf slope or the Pareto exponents. The only difference is a minor increase in the Pareto MLE exponent. It does however influence the estimations of vocabulary growth, which are displayed in table 4.12. Thus vocabulary growth estimation might be a more sensitive measure than word frequency distributions.
Looking at the differences between the observed and expected intercepts, it seems that a decrease in quality actually improves the estimates. This is however misleading, since the total number of types, and thus the percentual difference in light of the whole text, is greater for the flawed version. When this is taken into account, the estimates made on the flawed corpus are slightly worse than those for the gold standard.
The lesser versions of both *Moby-Dick* and *Alice* were created by inserting spelling errors, which results in a greater difference in quality between the gold standards and the flawed corpora. When we look at *Moby-Dick*, we clearly see a decrease in the Zipf slope and an increase in the Pareto LR exponent when the quality deteriorates (table 4.13). In contrast to what we saw with *Karenina*, here the Pareto MLE exponent decreases with quality.
Another interesting difference with *Karenina* is that for the 91% version of the corpus the interpolated as well as both the extrapolated vocabulary growths provide better estimates of the observed growth than they do for the gold standard. The fZM extrapolation gives the best result. In case of the 84% version, the fZM also gives the best estimate. However, both extrapolations perform worse than on the gold standard. Interestingly enough, the interpolated estimate is closer to the observed growth than that of the gold standard.
In the Zipf and Pareto results for the different quality versions of *Alice* (table 4.15, we see the same trends as we did for *Moby-Dick*. The Zipf slope and Pareto MLE exponent decrease for the flawed corpora, while the Pareto LR exponent increases. In the case of the vocabulary growth, all estimates made on the lesser versions are better than those made on the gold standard. The interpolation gets better with the 92% version and then decreases slightly in quality with the 85% version, but it is still better than the gold standard.
The quality of the extrapolation increases when the corpus quality decreases

for both fZM and GIGP. The best result is obtained with a fZM extrapolation of the 85% version.

**Anna Karenina**

| | *Zipf slope* | *Pareto MLE* | *Pareto LR* |
|---|---|---|---|
| *100%* | 1.02 | 2.97 | 1.97 |
| *99.6%* | 1.02 | 2.98 | 1.97 |

**Table 4.11:** Zipf slopes and Pareto exponents for the gold standard and reduced quality versions of *Anna Karenina*. **Gold standard:** $N$ = 226,446; $V$ = 31,885. **99.6%:** $N$ = 219,810; $V$ = 27,553.

**Anna Karenina**

| | *observed* | *interpolated* | *extrapolated fZM* | *extrapolated GIGP* |
|---|---|---|---|---|
| *100%* | 1.34 (4302) | 1.34 (4722) | 1.34 (4637) | 1.34 (4673) |
| *99.6%* | 1.29 (4082) | 1.30 (4476) | 1.30 (4374) | 1.30 (4410) |

**Table 4.12:** Vocabulary growth exponents for the gold standard and reduced quality versions of *Anna Karenina*. The exponents were obtained by performing a linear regression on the growth curves. The intercepts of the regression lines are between parentheses. See table 4.11 for the number of tokens and types in each corpus.

## 4.5 British National Corpus

Table 4.17 shows the Zipf slopes and Pareto exponents for the entire British National Corpus, both tokenized and normalized. The Zipf slope and Pareto LR exponent deviate from the expected values of 1 and 2 for both versions. Interestingly enough, this deviation is greatest for the normalized version. The Pareto MLE exponent however, is very high in case of the tokenized version. Normalization leads to a lower Pareto MLE exponent, which is actually below the expected Pareto value of 2. This is the lowest Pareto MLE value we have seen so far.
The vocabulary growth results for *Anna Karenina*, *Moby-Dick* and *Alice* did not prove to be a good measure for corpus quality, as we will expand upon in section 5.1, therefore we will not report them for the BNC and instead focus only on the Zipf and Pareto results.
Since the corpus contains both written documents and transcriptions of speech and each document differs in size, the corpus is very fragmented. When we process the corpus as one whole text, these differences all influence the result, but it is unclear in what way. Therefore we split the corpus

**Moby-Dick**

| | *Zipf slope* | *Pareto MLE* | *Pareto LR* |
|---|---|---|---|
| *100%* | 1.00 | 2.88 | 2.00 |
| *91%* | 0.95 | 2.85 | 2.05 |
| *84%* | 0.94 | 2.84 | 2.06 |

**Table 4.13:** Zipf slopes and Pareto exponents for the gold standard and reduced quality versions of *Moby-Dick*. **Gold standard:** $N$ = 217,852; $V$ = 38,638. **91%:** $N$ = 217,852; $V$ = 40,148. **84%:** $N$ = 217,852; $V$ = 40,546.

**Moby-Dick**

| | *observed* | *interpolated* | *extrapolated* | *extrapolated* |
|---|---|---|---|---|
| | | | *fZM* | *GIGP* |
| *100%* | 1.46 (4531) | 1.46 (5105) | 1.46 (4851) | 1.46 (4893) |
| *91%* | 1.48 (4845) | 1.48 (5387) | 1.48 (5166) | 1.48 (5209) |
| *84%* | 1.49 (4929) | 1.49 (5481) | 1.49 (5277) | 1.48 (5324) |

**Table 4.14:** Vocabulary growth exponents for the gold standard and reduced quality versions of *Moby-Dick*. The exponents were obtained by performing a linear regression on the growth curves. The intercepts of the regression lines are between parentheses. See table 4.13 for the number of tokens and types in each corpus.

**Alice**

| | *Zipf slope* | *Pareto MLE* | *Pareto LR* |
|---|---|---|---|
| *100%* | 0.91 | 3.05 | 2.09 |
| *92%* | 0.86 | 3.01 | 2.15 |
| *85%* | 0.82 | 2.98 | 2.20 |

**Table 4.15:** Zipf slopes and Pareto exponents for the gold standard and reduced quality versions of *Alice*. **Gold standard:** $N$ = 30,404; $V$ = 6,754. **92%:** $N$ = 30,404; $V$ = 7,246. **85%:** $N$ = 30,404; $V$ = 7,370.

**Alice**

| | *observed* | *interpolated* | *extrapolated* | *extrapolated* |
|---|---|---|---|---|
| | | | *fZM* | *GIGP* |
| *100%* | 1.57 (760) | 1.59 (945) | 1.60 (921) | 1.59 (933) |
| *92%* | 1.63 (838) | 1.65 (1024) | 1.65 (998) | 1.65 (1016) |
| *85%* | 1.64 (864) | 1.66 (1061) | 1.67 (1018) | 1.66 (1037) |

**Table 4.16:** Vocabulary growth exponents for the gold standard and reduced quality versions of *Alice*. The exponents were obtained by performing a linear regression on the growth curves. The intercepts of the regression lines are between parentheses. See table 4.15 for the number of tokens and types in each corpus.

into pieces of 250,000 tokens each. Figures 4.1 and 4.2 show the Zipf slopes, Pareto MLE exponents and Pareto LR exponents for all 452 pieces. In fact there were 453 pieces after splitting the corpus, but the final piece was smaller than the rest. We decided to remove this part to make sure corpus size would not be a factor in the results.

The graphs clearly show that normalization does not only decrease the peaks in the results, but also leads to an overall lower Pareto MLE exponent. In the tokenized version the average value is 2.41 with a standard deviation of 0.22. In the normalized version the average lowers to 2.05 with a standard deviation of only 0.09.

Normalization does not make such a big difference for the Zipf slope and Pareto LR exponent. With tokenization the average values are 0.99 and 2.00, both with a standard deviation of 0.04. These results are in line with those reported by Ferrer i Cancho and Solé [2]. Normalization only seems to influence the average Zipf slope, which averages at 1.00, again with a standard deviation of 0.04. The Pareto LR exponent has the same average value of 2.00, also with a standard deviation of 0.04.

Of course, these results are exactly what we want to see for a high quality corpus like the BNC. But to what extent are they influenced by the written or spoken parts of the corpus? In order to answer that question we divided the tokenized version into the original documents contained in the corpus. We ran our tests on each piece and compared the results for the written part with those for the spoken part.

Figure 4.3 presents the Zipf and Pareto results for all documents in the corpus, both written and spoken. We immediately see that the Zipf slope is a lot lower when measured on each separate document. It averages at $0.89 \pm 0.09$ and the Pareto LR exponent rises to an average of $2.12 \pm 0.15$. The Pareto MLE exponent shows the most variation with an average of $2.89 \pm 0.75$.

There are a lot of extreme values that are shown in the graph as high peaks. These are outliers in the data and have a disproportionate influence on the average values. We determined the outliers by using the interquartile range method[1] on the Zipf slopes and removed them from the results. The results after outlier removal are displayed in figure 4.4. Here the Zipf slope averages at $0.91 \pm 0.06$, the Pareto LR exponent at $2.09 \pm 0.08$ and the Pareto MLE exponent at $2.80 \pm 0.58$. The average number of tokens per document is 29,004.

The upper outliers, 11 in total, are all from the spoken part of the BNC. Their average Zipf slope is $1.15 \pm 0.10$ with a Pareto LR exponent of $1.84 \pm$

[1]http://www1.hollins.edu/faculty/clarkjm/Stat140/Outliers.htm

0.09. The average number of tokens per document is 81,586, which is much higher than the average size of the inliers. There are also 253 lower outliers. Their average Zipf slope is 0.63 ± 0.07 with a Pareto LR exponent of 2.55 ± 0.23. Here the average document size is only 1,368 tokens. The lower outliers are formed by most of the smallest documents in the BNC. 86 of them (34%) are from the spoken part. In total, 10.6% of the spoken part is in the outliers.

When we separate the written documents from the spoken ones, we see that the average of the Zipf slope for the written part is the same as it is for both parts together: 0.89 ± 0.09. The Pareto LR exponent also averages at the same value, 2.12, but with a slightly lower standard deviation of 0.14. For the spoken part however, the Zipf slope decreases slightly to an average value of 0.88 ± 0.11. The Pareto LR exponent consequently increases to a value of 2.14 ± 0.16. The Pareto MLE exponent is highest for the written part with an average of 3.03 ± 0.73. It averages at 2.39 ± 0.60 for the spoken part.

Again, we applied the interquartile range method to filter out any outliers for both parts. Figure 4.5 shows the results for the inliers of the written part. The results for the inliers of the spoken part are displayed in figure 4.6. Removing the outliers results in a higher Zipf slope and lower Pareto exponent for both portions of the BNC, with values of 0.91 ± 0.06 and 2.09 ± 0.07 respectively for the written part and 0.89 ± 0.08 and 2.12 ± 0.11 for the spoken part. The Pareto MLE exponent shows a slight decrease in both cases. For the written part it centers at 2.93 ± 0.56, for the spoken part at 2.31 ± 0.11.

The written part of the BNC only has lower outliers, with an average Zipf slope and Pareto exponent of 0.64 ± 0.08 and 2.56 ± 0.24 respectively. The average number of tokens is 2,228, which is extremely low compared to the average of 33,575 tokens for the inliers. 47.6% of the outliers are collections of newspaper articles. Newspapers are characterized by many short articles on diverse topics and written by different authors. When these articles are processed together, this results in an extremely incoherent text. This, together with the small size of the documents, explains the low Zipf slope and high Pareto LR exponent.

The spoken part has both upper and lower outliers. Again, the lower outliers are formed by the smallest documents, with an average size of merely 342 tokens. The Zipf slope averages at 0.61 ± 0.07 with a Pareto LR exponent of 2.60 ± 0.27. The average size of the upper outliers is 60,205, which is almost five times as high as the average size of the inliers, where $N$ is 12,658. The Zipf slope and Pareto LR exponent of the upper outliers are 1.20 ± 0.12 and 1.79 ± 0.10 respectively.

## British National Corpus

| | *Zipf slope* | *Pareto MLE* | *Pareto LR* |
|---|---|---|---|
| *tokenized* | 1.11 | 3.09 | 1.89 |
| *normalized* | 1.15 | 1.74 | 1.85 |

**Table 4.17:** Zipf slopes and Pareto exponents for the tokenized and normalized versions of the BNC. Tokenized: $N$ = 113,169,394; $V$ = 833,944. Normalized: $N$ = 114,417,984; $V$ = 519,931.

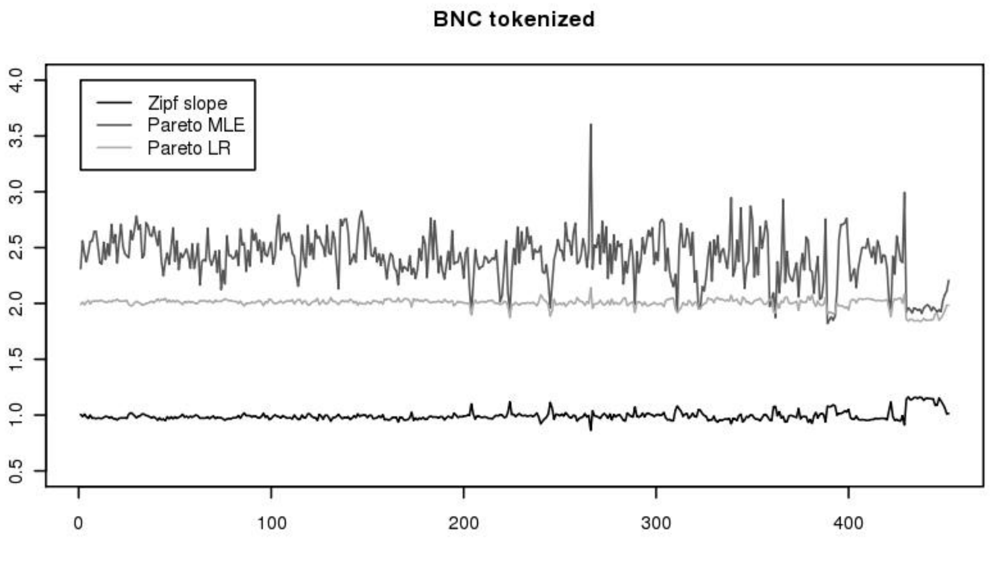

**Figure 4.1:** Slopes and exponents for the tokenized version of the BNC

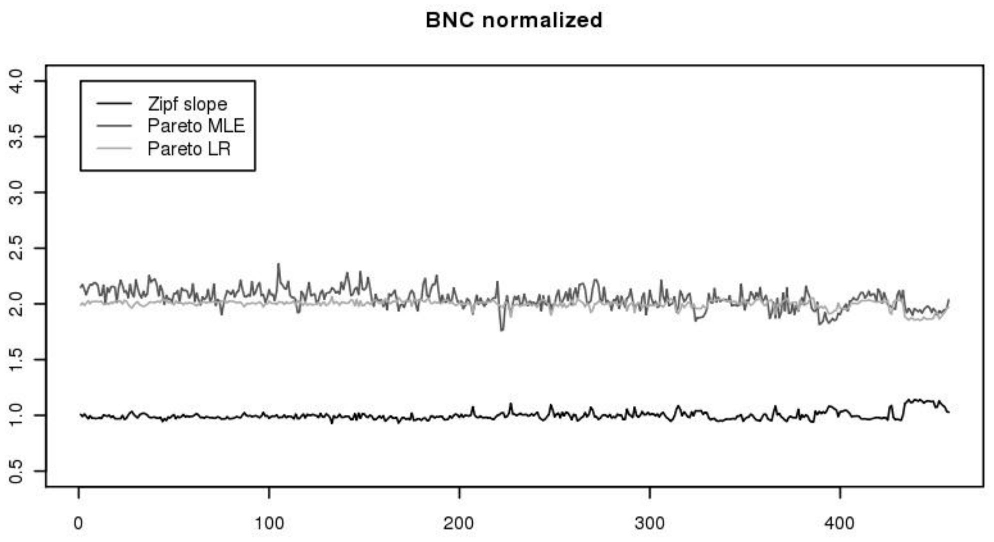

**Figure 4.2:** Slopes and exponents for the normalized version of the BNC

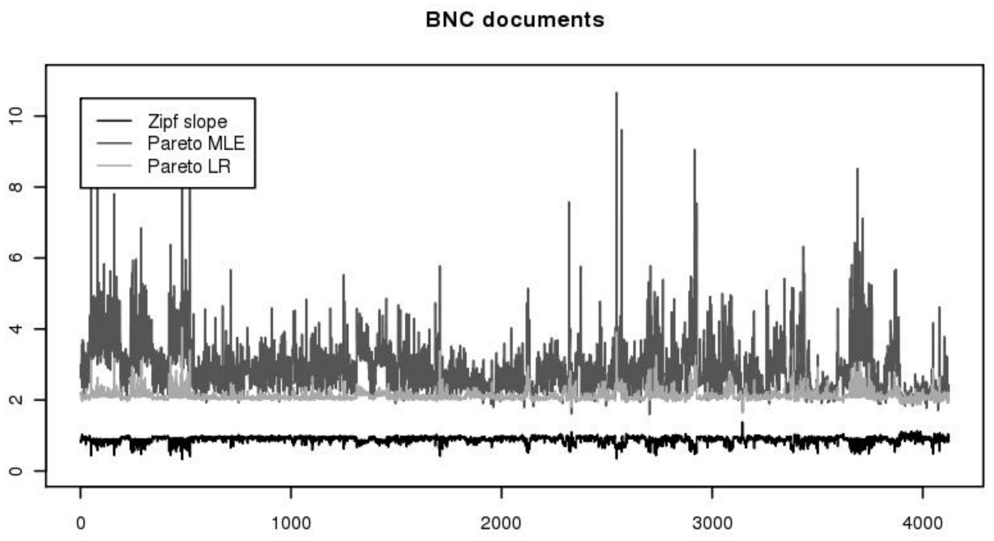

**Figure 4.3:** Slopes and exponents for the documents in the BNC

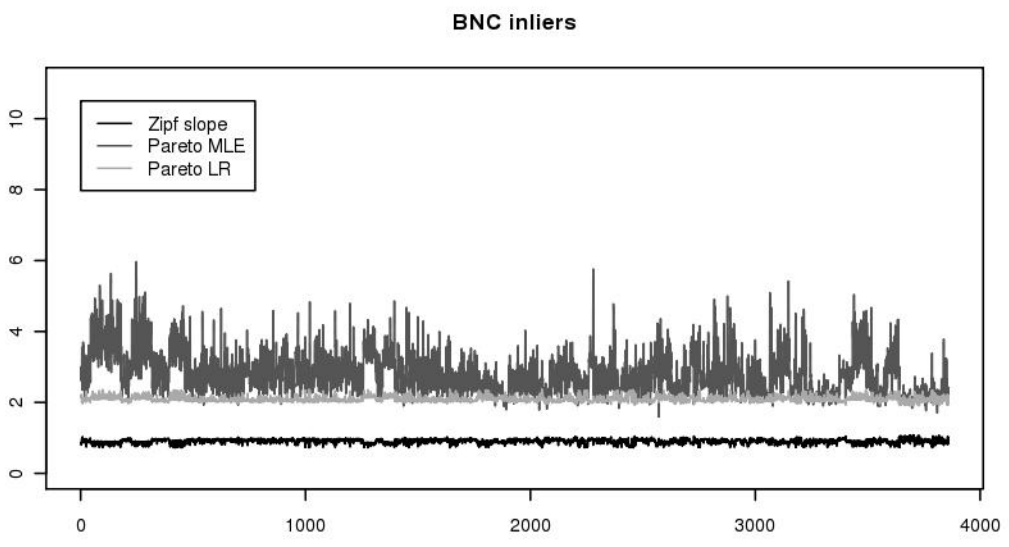

**Figure 4.4:** Slopes and exponents for the inliers of the documents in the BNC. Inliers are determined by applying the interquartile range method to the Zipf slopes.

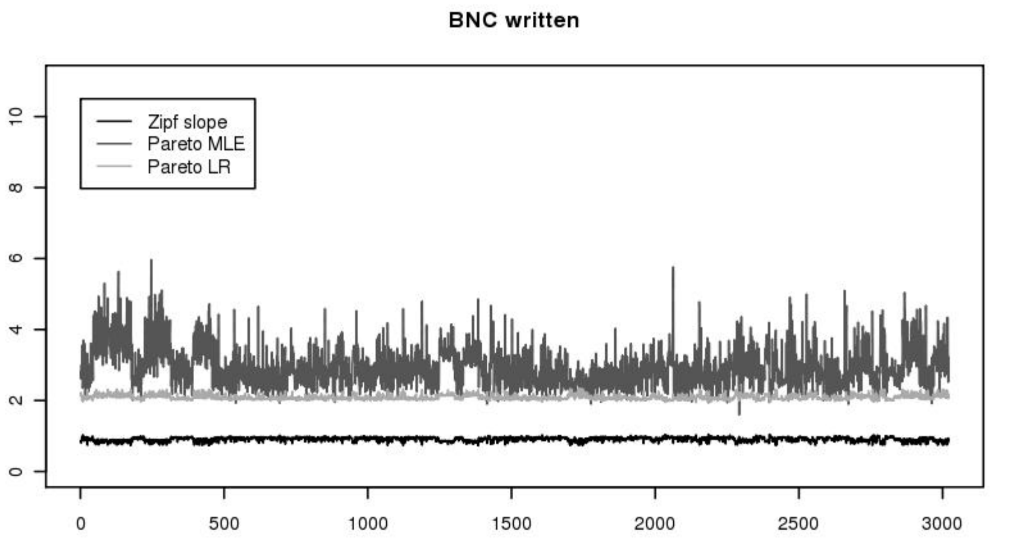

**Figure 4.5:** Slopes and exponents for the inliers of the written documents in the BNC. Inliers are determined by applying the interquartile range method to the Zipf slopes.

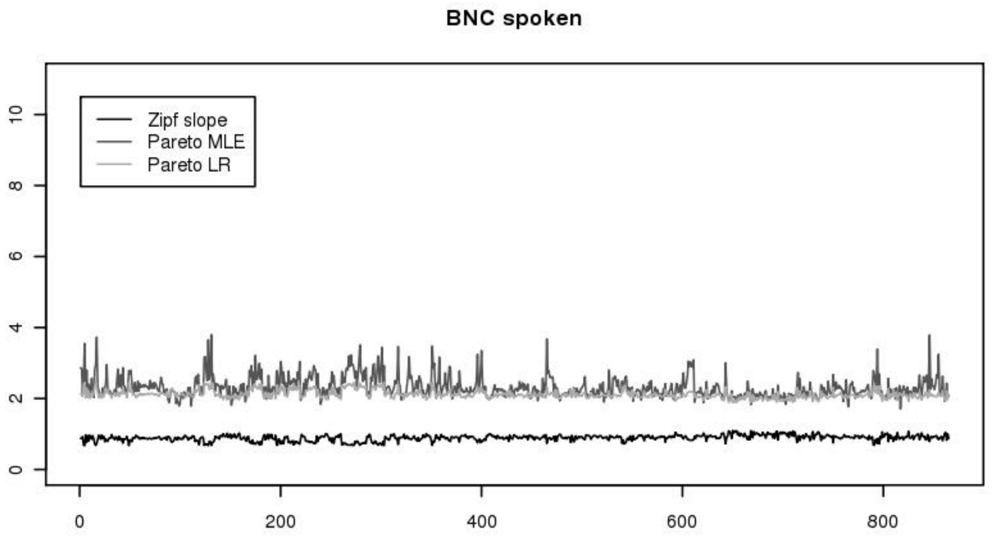

**Figure 4.6:** Slopes and exponents for the inliers of the spoken documents in the BNC. Inliers are determined by applying the interquartile range method to the Zipf slopes.

# Chapter 5

# Discussion

## 5.1 Vocabulary growth

The exponents of the regression lines fitted to the vocabulary growth data do not have steady values we can draw conclusions from. This makes it very difficult, if not impossible, to use them to determine corpus quality. The only clear observation we can make is that the exponents go down when tokenization is applied to the text. This implies that the vocabulary grows at a slower pace in a tokenized corpus. An explanation for this phenomenon is given by the fact that words which have a punctuation mark attached to them in an untokenized text count as a separate type and thus increase the vocabulary with an extra item.
Tokenization does also create an extra vocabulary entry by splitting the punctuation mark from its preceding or following word, but all appearances of the same punctuation mark will count as only one type with a higher frequency, while all words with punctuation marks attached will most probably appear only once in that fashion. They are then counted as hapaxes and this increases the total number of types at a much higher rate than concatenated punctuation does.
This argument also holds true for the transformation to lower case, though not to such extremes. This is shown by the minor decrease of the exponent when lower case is applied.
The intercepts of the regression lines do give us more insight into the quality of the fit: the closer the intercept of the expected growth is to the intercept of the observed growth, the better the fit. But they still do not tell us anything about the quality of the corpus itself. The raw versions of *Anna Karenina* and *Moby-Dick* provide the best fits, while tokenization seriously improves the fit for *Alice*. This is very difficult to explain in light of the used corpora.

*Alice* is not only ten times smaller than both other corpora, it is also of a completely different genre and far more fragmented in its structure. All of these factors could be of influence on the data. We tried to determine what role corpus size plays in this by decreasing the size of *Anna Karenina* and *Moby-Dick*, but the results show no common trends.
Another aspect which makes it difficult to draw any conclusions based on estimations of vocabulary growth is that they use a sample of the empirical data. If we were to run the tests again other data points may be sampled and the results could be totally different.

## 5.2 Zipf slope and Pareto LR exponent

As we mentioned in section 2.1 much of the literature reports a Zipf slope of 1 for word frequency distributions and a Pareto exponent of 2 [2, 4, 5, 8]. The results for the raw, high quality versions of *Anna Karenina* and *Moby-Dick*, and also for the BNC split into equal parts of 250,000 tokens, confirm this.
Several aspects seem to influence these values though. According to our results, a smaller corpus size leads to a lower Zipf slope and a higher Pareto LR exponent. This is seen not only in the results for our smallest corpus, *Alice*, but also in the results for *Karenina* and *Moby-Dick* when they are reduced in size. When we run the tests on all separate documents of the BNC, with an average size of approximately 29,000 tokens, we see the same change in both the Zipf and Pareto LR results. The Zipf slope averages at 0.91 for the written part of the BNC, the same value as we obtained for *Alice*, which is of roughly the same size at 30,000 tokens. The Pareto LR exponents are also the same at 2.09.
More so, the Zipf slope for the BNC as a whole actually increases to values over 1.10, with a Pareto LR exponent below 1.90. Our tokenized version of the BNC contains around 113 million tokens, which is a size several orders of magnitude larger than any of the other corpora we used.
As explained in section 5.1, tokenization increases the number of tokens. The results for *Karenina* and *Moby-Dick* also show an increase in the Zipf slope when the corpus is tokenized. Transforming everything to lower case has the same effect, although the number of tokens does not increase. The number of types however decreases, as it does with tokenization or normalization. Thus the number of types, $V$, in a text must be of influence to the values of Zipf and Pareto. This is easily explained by the fact that the Zipf distribution is a distribution of the frequencies of types, not tokens.
Since the number of types decreases with lower case and the number of tokens stays the same, some types will have a higher frequency and receive a higher

rank (closer to 1) in the Zipf distribution. A higher rank means the type will move more to the left of the log-log plot towards the y-axis. When more types receive higher ranks, the plotted line gets steeper and the Zipf slope increases.
In terms of the Pareto law, when more types have higher frequencies, the probability for the lower frequencies decreases, making the log-log plot less steep and the Pareto exponent lower.
Oddly enough, when tokenization is applied to *Alice*, the Zipf slope goes down and the Pareto LR exponent goes up, instead of the other way around. This means that the number of types is increased rather than decreased by tokenization. This could be caused by the use of many different punctuation marks in the text. Another explanation is that because of the small size of the corpus, the number of words that appear next to the punctuation marks are few or often the same. If this is the case, then tokenization unnecessarily adds a lot of types in the form of punctuation marks, more than the few punctuated words. To verify this the text should be investigated in more detail.
The results we obtained for the lower quality versions of *Moby-Dick* and *Alice* show a decrease in Zipf slope and an increase in Pareto LR exponent. For both corpora, a decrease in quality of 7-8% leads to a Zipf slope that is 0.05 lower and a Pareto LR exponent that is 0.05-0.06 higher. However, when the quality of the corpus decreases twice as much, the difference in the Zipf and Pareto results does not automatically double. This is an artifact of our method, since the list of misspellings used is limited. After a certain point no new types are added to the text, because more of the same words are replaced by the same typos.
The decrease in quality was created by inserting spelling errors in the corpus. This does not influence the total number of tokens, but it does add a lot of extra types with lower frequencies and consequently lower ranks. Visually speaking, they move more to the right of the x-axis on the Zipf plot. Also, the frequencies of the higher ranked words slightly decrease, since some of them are misspelled. This results in a log-log plot that is less steep and so the Zipf slope decreases. Coincidentally, the lower frequencies receive higher probabilities in the Pareto distribution, leading to a higher Pareto LR exponent.

## 5.3 Pareto MLE exponent

The Pareto exponent computed using maximum likelihood does not produce the results we expected. Clauset, Shalizi and Newman report that this method takes precedence over the linear regression method, saying that it produces more reliable results [13]. They report a Zipf slope of 0.90 and a Pareto MLE exponent of 1.95 for *Moby-Dick*. However, they do not provide sufficient details on the version or the preprocessing of the text for us to reproduce these results. Looking at the results we obtained on our data, the linear regression method seems to outperform the maximum likelihood method.

Our results for the Pareto MLE exponent do not clearly center around a certain value. The only result we obtained that was anywhere near the expected value of 2 was for the normalized version of the BNC. Here the Pareto MLE exponent was 1.74, which is way below the expected value of 2. All other measurements were well above 2 and often even above 3.

The only concrete thing we can say based on our results is that tokenization and lower case transformation decrease the exponent for the same reasons as they decrease the exponent obtained with the linear regression method. In case of a decrease in corpus quality however, the MLE exponent goes down, while the LR exponent goes up. As explained in section 5.2, our flawed corpora contain more types with low frequencies. These low frequencies thus get higher probabilities, resulting in a higher exponent. Since the Pareto MLE exponent does not show this same logical transformation, we suspect that there is something wrong with the implementation of this method in the software provided by Clauset [13].

# Chapter 6

# Conclusions and Recommendations

We set out to find a metric for the unsupervised determination of corpus quality. Unfortunately we did not find a solid metric. We did however discover a crude measure in the Zipf slope and Pareto exponent, which gives a global indication of the quality of a corpus. A decrease in corpus quality leads to a lower Zipf slope and a higher Pareto exponent, as is shown by the results we obtained on *Moby-Dick* and *Alice*. These differences in slope and exponent are related to one another as explained by the equations given by Ferrer i Cancho [2].

The values of 1 and 2 reported in the literature hold true for corpora around the size of 250,000 tokens [2, 4, 5, 8]. Adjusting the corpus size affects these values though. For larger corpora the Zipf slope increases, while the Pareto exponent decreases. It works the other way around for smaller corpora. It is worth investigating if different text sizes have different standard values.

Montemurro obtained his results by processing 2,606 English books from Project Gutenberg as one whole text [4]. We are very interested to see what will happen when our power_exploRe script is run on all these texts separately and their Zipf slopes and Pareto exponents are averaged. This way the illusion of a single author is kept and differences in topic or choice of words are normalized.

Moreover, Gelbukh and Sidorov report that the value of the Zipf slope also depends on the language used [3]. More inflective instead of analytical languages like Russian and Spanish lead to a lower Zipf slope, and consequently a higher Pareto exponent. Dutch is far more inflective than English. We would therefore expect to see a lower Zipf slope in the results for *Anna Karenina*, but this is not the case.

Contrary to what we expected, the estimated vocabulary growth data did

not provide a clear insight into what changes in a corpus when the quality decreases. This does not necessarily mean that vocabulary growth is useless for determining corpus quality. We compared the observed and estimated growths by performing a linear regression on them and comparing the slopes and intercepts of the regression lines. Vocabulary growth follows a curve rather than a straight line. Performing a linear regression on a nonlinear graph does not take into account the smallest and most subtle changes in the curve, which is exactly what we want to measure. Therefore, the chosen method does not contribute to our goal. In light of the results we obtained for the Zipf and Pareto distributions, we need to re-examine our data with a closer eye on the type-token and hapax-type ratios.
A more elaborate method of assessing corpus quality may be found in the field of complex networks. Antiqueira et al. [12] have made great progress in this area by measuring the correlations between the quality of text content and a range of network features. They found a strong correlation between text quality and coherence, which shows that looking at a text as a network of concepts can reveal shortcomings in its structure.
The properties of network features have in recent years become far better understood through advances in complexity and small-world theory. Antiqueira et al. [12] track the importance of connected components in the textual network. Ferrer i Cancho [10] shows that the largest connected component is directly linked to the size of the Pareto exponent. Throughout the present work we reaffirm the direct link between the Pareto exponent and the Zipf slope. Measuring more detailed network features, as in [12], is far more time consuming than measuring either the Zipf slope or the Pareto exponent.
Further, Ferrer i Cancho [10] cogently argues that:

1. most criticisms of Zipf's law have not taken into account the exponent;

2. the exponent of Zipf's law in single author text lies between 1.6 and 2.4, as we confirm in the present work;

3. a large Pareto exponent is not impossible in natural language, but it is unlikely to be found in a system combining words through semantic constraints;

4. up to now the largest values of the Pareto exponent have only been found in single author text samples obtained from schizophrenic patients in the acute phase of the illness. These texts display disturbances in structure, coherence and organization of thought, leading to reduced intelligibility and speech disorganization, resulting in greater difficulty or even impossibility of comprehension.

We have observed a limited increase in the Pareto exponent by inserting typos in our flawed corpora. While this does not effect that “language breaks into pieces”, we have nevertheless seen this happen in badly OCRed corpora. We hope to one day find a more accurate measure for this phenomenon.

# Appendix A

# Software Used

### ABBYY Finereader

ABBYY Finereader is one of the most widely used commercial OCR packages today. It is developed by the Russian company ABBYY (www.abbyy.com). We used two versions of the software to convert scanned images into editable text: 8.0 Professional Edition and 9.0 Professional Edition. According to the ABBYY company, the main differences between versions 8.0 and 9.0 are: automatic language detection, enhanced recognition of document structure and better accuracy.[1]

### compute_emp_vgc.pl

`compute_emp_vgc.pl` is an open source Perl program written by Baroni and Evert [14]. It is used to compute the observed vocabulary growth, i.e. the number of types, and the number of hapax legomena of a text file in one-token-per-line format. The program requires the interval at which to compute the growth to be defined. For instance, if the argument given to the program is 100, *V* (the number of types) and *V1* (the number of hapaxes) will be measured after every 100 tokens in the text. The resulting vocabulary growth files can be imported into the R statistical environment to create vocabulary growth curves using the zipfR package. The program can be downloaded from the zipfR website.[2]

[1]http://finereader.abbyy.com/?param=137516

[2]http://www.cogsci.uni-osnabrueck.de/∼severt/zipfR/

### ILK Tokenizer

The ILK Tokenizer is a Perl program developed by the Induction of Linguistic Knowledge research group at Tilburg University.[3] It splits a text into separate tokens by searching for word and sentence boundaries and can be used for a variety of languages.

### languageR

The languageR package for R is developed by R.H. Baayen. Besides exemplary data sets used in his book *Analyzing Linguistic Data: A practical introduction to statistics using R* [15], the package contains functions for statistical lexical analyses. The open source package is available on the Comprehensive R Archive Network website.[4]

### OCR Frontiers Toolkit

The OCR Frontiers Toolkit is a set of Unix tools for determining the accuracy of OCR output against a gold standard text. The toolkit is a supplement to the book *Optical Character Recognition: An Illustrated Guide to the Frontier*, by Rice, Nagy, and Nartker [19].

### pareto.R

Clauset provides some useful software regarding power laws on his website.[5] The `pareto.R` package is a collection of several R functions for estimating continuous Pareto power laws. It is a companion to the article *Power-law distributions in empirical data* by Clauset, Shalizi and Newman [13].

### R

R is an environment for performing statistical analyses and creating graphs. The software is open source and can be obtained through the Comprehensive R Archive Network website.[6]

### randomreplace.10.pl

`randomreplace.10.pl` is a Perl program written by Reynaert [18]. It randomly replaces words in a one-token-per-line formatted text by a misspelled

[3]http://ilk.uvt.nl/

[4]http://cran.r-project.org/

[5]http://www.santafe.edu/ aaronc/powerlaws/

[6]http://cran.r-project.org/

variant of the same word. The words and their misspellings are to be listed in a separate file, which can be defined at the time of execution together with the desired amount of replacements in percentages.

### ScanSoft OmniPage

ScanSoft OmniPage is a product of the company Nuance, based in the United States (www.nuance.com). OmniPage is a high-end commercial OCR package. We used the Professional edition of OmniPage version 16 to convert scanned images into editable text.

### zipfR

ZipfR is an open source package for the R statistical environment. The package is developed by Baroni and Evert [14] and is used for visualizing vocabulary growth and LNRE modeling of word frequency distributions. The open source package is available on the Comprehensive R Archive Network website.[7]

[7]http://cran.r-project.org/

# Appendix B

# Power exploRe

## B.1 power_exploRe.sh

```
#!/bin/sh

## Call: sh power\_exploRe.sh [dir] [vgc-interval]
## [dir] is the directory path where the text files are located
## (without the trailing slash).
## [vgc-interval] is the interval at which the vocabulary growth
## is measured by the compute_emp_vgc.pl program.
## The texts should be in a one-token-per-line format.
## The script depends on several other scripts/programs/files:
## - f.txt: a text file containing only the letter f on the first
## line. The script expects the file to be located in the working
## directory.
## - compute_emp_vgc.pl: Perl program developed by Evert and Baroni
## (2007). The script expects the program to be located in the
## working directory.
## - power\_exploRe.R: an R program that calculates the vocabulary growth,
## the Zipf slopes and the Pareto exponents. The script expects the
## program to be located in the working directory.
## - pareto.R: an R program to calculate the Pareto distribution
## provided by Clauset (2007). The script expects the program to be
## located in the working directory.
##
## OUTPUT
## The script creates a separate folder in [dir] for every pro-
## cessed text. The folder contains:
## - a copy of the text
## - the vgc-file generated by compute_emp_vgc.pl
## - a term frequency list (.tfl)
## - a postscript file containing plots of the vocabulary growth
## and the Zipfian ditribution
## - R.out (output from fabriek.R)
## - R.new (reformatted version of R.out)
## - R.stderr (error messages generated by fabriek.R)
## [dir] contains 'data.txt', which lists the results for all pro-
## cessed texts in this order: Zipf slope, Pareto MLE exponent, Pa-
## reto LR exponent, observed growth intercept, slope and exponent,
## interpolated growth intercept, slope and exponent, fZM intercept,
## slope and exponent, GIGP intercept, slope and exponent, N(obser-
## ved), V(observed), V1(observed)

ls $1/*.txt > list

( while read x;
## generate term frequency lists
do sort < $x | uniq -c | sort -nr | sed 's/^\s*//' | tr -s " " "\t" |
+ cut -f1 > ${x%.*}.tf
paste -s -d"\n" f.txt ${x%.*}.tf > ${x%.*}.tfl
rm ${x%.*}.tf
## generate vgc files
perl compute_emp_vgc.pl -l $2 $x > ${x%.*}.vgc
TEXT=$x
VGC=${x%.*}.vgc
TFL=${x%.*}.tfl
## export variables to R
export TEXT
export VGC
export TFL
mkdir ${x%.*}
## call R program
R --slave < power\_exploRe.R > ${x%.*}/R.out 2> ${x%.*}/R.stderr --no-save
cp $x ${x%.*}
mv ${x%.*}.vgc ${x%.*}/
mv ${x%.*}.tfl ${x%.*}/
mv plot.ps ${x%.*}/;
done ) < list
rm list

ls $1/*/R.out > outlist

## remove unwanted data from R.out
```

```
( while read x;
do sed '/..\+/d' < $x > ${x%.*}.tmp
sed '/null device /d' < ${x%.*}.tmp > ${x%.*}.tmp2
sed '/          1 /d' < ${x%.*}.tmp2 > ${x%.*}.tmp
sed 's/\[1\] //g' < ${x%.*}.tmp > ${x%.*}.tmp2
sed 's/Call://g' < ${x%.*}.tmp2 > ${x%.*}.tmp
sed 's/Coefficients://g' < ${x%.*}.tmp > ${x%.*}.tmp2
sed 's/(Intercept)  N(.*)  //g' < ${x%.*}.tmp2 > ${x%.*}.tmp
sed 's/lm(formula = V(.*) ~ N(.*))//g' < ${x%.*}.tmp > ${x%.*}.tmp2
sed 's/^ \+//g' < ${x%.*}.tmp2 > ${x%.*}.tmp
sed 's/ \+$//g' < ${x%.*}.tmp > ${x%.*}.tmp2
sed 's/ \+/\t/g' < ${x%.*}.tmp2 > ${x%.*}.tmp
tr "." "," < ${x%.*}.tmp | tr -s "\n" "\t" | tr -s "\s" "\t" > ${x%.*}.new
rm ${x%.*}.tmp
rm ${x%.*}.tmp2;
done ) < outlist

## put data from all processed texts into 1 tab-delimited text file
ls $1/*/R.new > outlist
tr -s "\n" " " < outlist > $1/files
sed 's/^/paste -s -d "\\n" /' < $1/files > $1/tmp
sed 's/$/> data.txt/g' < $1/tmp > $1/script.sh
sh $1/script.sh
mv data.txt $1/
rm $1/tmp
rm outlist
rm $1/files
rm $1/script.sh
```

## B.2 power_exploRe.R

```
# load required libraries (can be downloaded from http://cran.r-project.org/)
library(zipfR)
library(languageR)

# import variables from shell script
TEXTtxt <- Sys.getenv("TEXT")
TEXTvgc <- Sys.getenv("VGC")
TEXTtfl <- Sys.getenv("TFL")

# read text
TEXT.txt <- scan(TEXTtxt, what="character")
```

```
# read vgc file
TEXT.vgc <- read.vgc(file(TEXTvgc))

# read term frequency list
TEXT.tfl <- read.tfl(file(TEXTtfl))

# transform term frequency list to frequency spectrum
TEXT.spc <- tfl2spc(TEXT.tfl)

# compute interpolated growth for entire text size from frequency spectrum,
# including hapaxes
TEXT.bin.vgc <- vgc.interp(TEXT.spc, N(TEXT.vgc), m.max=1)

# compute extrapolated growth for entire text size from a sampled frequency
# spectrum, including hapaxes, using LNRE models fZM and GIGP
TEXT.ext.spc <- sample.spc(TEXT.spc, N=(N(TEXT.spc)))
TEXT.fzm <- lnre("fzm", TEXT.ext.spc, exact=FALSE)
TEXT.fzm.vgc <- lnre.vgc(TEXT.fzm, N=N(TEXT.vgc), m.max=1)
TEXT.gigp <- lnre("gigp", TEXT.ext.spc, exact=FALSE)
TEXT.gigp.vgc <- lnre.vgc(TEXT.gigp, N=N(TEXT.vgc), m.max=1)

# perform linear regression on observed, interpolated and both extrapolated
# growths
TEXT.fit <- lm(V(TEXT.vgc)~N(TEXT.vgc))
TEXT1.fit <- lm(Vm(TEXT.vgc,m=1)~N(TEXT.vgc))
TEXT.bin.fit <- lm(V(TEXT.bin.vgc)~N(TEXT.bin.vgc))
TEXT1.bin.fit <- lm(Vm(TEXT.bin.vgc,m=1)~N(TEXT.bin.vgc))
TEXT.fzm.fit <- lm(V(TEXT.fzm.vgc)~N(TEXT.fzm.vgc))
TEXT1.fzm.fit <- lm(Vm(TEXT.fzm.vgc,m=1)~N(TEXT.fzm.vgc))
TEXT.gigp.fit <- lm(V(TEXT.gigp.vgc)~N(TEXT.gigp.vgc))
TEXT1.gigp.fit <- lm(Vm(TEXT.gigp.vgc,m=1)~N(TEXT.gigp.vgc))

# calculate exponent for each fit by taking 10 to the power of the slope of
# the regression line
TEXT.exp <- 10^TEXT.fit$coefficients[[2]]
TEXT1.exp <- 10^TEXT1.fit$coefficients[[2]]
TEXT.bin.exp <- 10^TEXT.bin.fit$coefficients[[2]]
TEXT1.bin.exp <- 10^TEXT1.bin.fit$coefficients[[2]]
TEXT.fzm.exp <- 10^TEXT.fzm.fit$coefficients[[2]]
TEXT1.fzm.exp <- 10^TEXT1.fzm.fit$coefficients[[2]]
TEXT.gigp.exp <- 10^TEXT.gigp.fit$coefficients[[2]]
TEXT1.gigp.exp <- 10^TEXT1.gigp.fit$coefficients[[2]]
```

```
# compute Zipfian distribution and perform a linear regression on it
TEXT.z <- zipf.fnc(TEXT.txt, plot=FALSE)
TEXT.z.fit <- lm(log(TEXT.z$frequency)~log(TEXT.z$rank))

# compute Pareto distribution and exponents using the maximum likeli-
# hood and linear
# regression methods
source("pareto.R")
# change to location of pareto.R if it is not in the working directory
TEXT.table = table(TEXT.txt)
TEXT.table = sort(TEXT.table, decreasing = TRUE)
TEXTtable <- pareto.fit(TEXT.table,threshold=1,method="ml")
TEXTtable2 <- pareto.fit(TEXT.table,threshold=1,method="regression.cdf")

# create plots
postscript("plot.ps")
plot(TEXT.vgc,TEXT.bin.vgc,TEXT.fzm.vgc,TEXT.gigp.vgc,add.m=1,bw=TRUE,
+ main="TEXT",legend=c("observed","interpolated","extrapolated fZM",
+ "extrapolated GIGP"))
plot(N(TEXT.vgc),V(TEXT.vgc),main="Observed growth")
abline(TEXT.fit)
plot(N(TEXT.bin.vgc),V(TEXT.bin.vgc),main="Interpolated growth")
abline(TEXT.bin.fit)
plot(N(TEXT.fzm.vgc),V(TEXT.fzm.vgc),main="Extrapolated growth fZM")
abline(TEXT.fzm.fit)
plot(N(TEXT.gigp.vgc),V(TEXT.gigp.vgc),main="Extrapolated growth GIGP")
abline(TEXT.gigp.fit)
plot(log(TEXT.z$rank),log(TEXT.z$frequency),type="S")
abline(TEXT.z.fit,col="darkgrey")
dev.off()

# write output
TEXT.z.fit$coefficients[[2]]
TEXTtable$exponent[[1]]
TEXTtable2$exponent[[1]]
TEXT.fit
TEXT.exp
TEXT.bin.fit
TEXT.bin.exp
TEXT.fzm.fit
TEXT.fzm.exp
TEXT.gigp.fit
TEXT.gigp.exp
```

```
tail(N(TEXT.vgc),n=1)
tail(V(TEXT.vgc),n=1)
tail(Vm(TEXT.vgc, m=1),n=1)
```

# Bibliography

[1] D.M.W. Powers. Applications and explanations of zipf's law. In *NeMLaP3/CoNLL98: New Methods in Language Processing and Computational Natural Language Learning*, pages 151–160. ACL, 1998.

[2] R. Ferrer i Cancho and R.V. Solé. Two regimes in the frequency of words and the origins of complex lexicons: Zipf's law revisited. *Journal of Quantitative Linguistics*, 8(3):165–173, December 2001.

[3] A. Gelbukh and G. Sidorov. Zipf and heaps laws coefficients depend on language. In *Proc. CICLing-2001, Conference on Intelligent Text Processing and Computational Linguistics*, pages 332–335, February 2001.

[4] M.A. Montemurro. Beyond the zipf-mandelbrot law in quantitative linguistics. *Physica A*, 300(3):567–578, November 2001.

[5] L.A. Adamic. Zipf, power-laws, and pareto - a ranking tutorial, 2002.

[6] S. Evert and M. Baroni. Testing the extrapolation quality of word frequency models. In *Proceedings of Corpus Linguistics 2005, Birmingham, UK*, volume 1 of *The Corpus Linguistics Conference Series*, July 2005.

[7] R. Ferrer i Cancho. Zipf's law from a communicative phase transition. *The European Physical Journal B*, 47(3):449–457, October 2005.

[8] M.E.J. Newman. Power laws, pareto distributions and zipf's law. *Contemporary Physics*, 46(5):323–351, October 2005.

[9] M. Baroni. Distributions in text. To appear in "Corpus linguistics: An international handbook", Berlin: Mouton de Gruyter, June 2006.

[10] R. Ferrer i Cancho. When language breaks into pieces: A conflict between communication through isolated signals and language. *Biosystems*, 84(3):242–253, June 2006.

[11] M.W.C. Reynaert. Corpus-induced corpus clean-up. In *LREC 2006: Fifth International Conference on Language Resources and Evaluation*, pages 87–92. ELRA: European Language Resources Association, May 2006.

[12] L. Antiqueira, M.G.V. Nunes, O.N. Oliveira Jr., and L. da F. Costa. Strong correlations between text quality and complex networks features. *Physica A: Statistical and Theoretical Physics*, 373:811–820, January 2007.

[13] A. Clauset, C.R. Shalizi, and M.E.J. Newman. Power-law distributions in empirical data, June 2007.

[14] S. Evert and M. Baroni. zipfR: Word frequency distributions in R. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, Posters and Demonstrations Session*, 2007.

[15] R. Harald Baayen. *Analyzing Linguistic Data: A Practical Introduction to Statistics using R.* Cambridge University Press, 2008.

[16] R. Harald Baayen. *Word frequency distributions*, volume 18 of *Text, Speech and Language Technology.* Kluwer, Dordrecht, 2001.

[17] D. Jurafsky and J.H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition.* Prentice Hall Series in Artificial Intelligence. Prentice Hall, 2nd edition, May 2008.

[18] M.W.C. Reynaert. *Text-Induced Spelling Correction.* Martin Reynaert, 2005.

[19] S.V. Rice, G.L. Nagy, and T.A. Nartker. *Optical Character Recognition: An Illustrated Guide to the Frontier.* Kluwer Academic Publishers, 1999.

[20] G.K. Zipf. *Human Behaviour and the Principle of Least Effort.* Addison-Wesley, Reading MA, 1949.