# Finite State Models for the Generation of Large Corpora of Natural Language Texts

D. Cantone, S. Cristofaro, S. Faro, and E. Giaquinta

Università di Catania, Dipartimento di Matematica e Informatica
Viale Andrea Doria 6, I-95125 Catania, Italy
{cantone | cristofaro | faro | giaquinta}@dmi.unict.it

Natural languages are probably one of the most common type of input for text processing algorithms. Therefore, it is often desirable to have a large training/testing set of input of this kind, especially when dealing with algorithms tuned for natural language texts. The problem in creating good corpora is that often natural language texts are too short with respect to the dimension required to test effectively the goodness of text processing algorithms, such as string matching and compression algorithms. This is, for instance, the case of the well-known Canterbury Corpus [RB97], used for testing lossless data compression algorithms, which contains natural language texts with a relative small dimension of not more than 500Kb. The only exception is the "King James Version of the Bible" (approximately $3,85$Mb) contained in the Large Corpus [RB97]. On the other hand corpora of non-textual data contain test files with dimensions up to 3Mb (like the Protein Corpus [NMW99] and the Silesia Corpus [D03]), while testing on random texts is often performed on buffers of dimension 10Mb [ACR99] and 20Mb [CF04].

In many cases the problem due to the lack of big corpus of natural language texts can be solved by simply concatenating a set of collected texts, even with heterogeneous contexts and by different authors. This is the case, for example, of The *Linguistic Data Consortium* (http://www.ldc.upenn.edu), an open consortium of universities which creates, collects and distributes speech and text databases and other resources for research and development purposes.

However, in this context, the task of being able to automatically generate texts which maintain properties of real texts is appealing. In this note we present a preliminary study on a finite state model for text generation which maintains statistical and structural characteristics of natural language texts, i.e., Zipf's law [Z32] and inverse-rank power law [CF03], thus providing a very good approximation for testing purposes.

## 1 Preliminaries

Before entering into details, we review a bit of notations and terminology. A string $S$ of length $m > 0$ is represented as a finite array $S[0 \,..\, m-1]$. The length of $S$ is denoted with $|S|$, i.e, $|S| = m$. By $S[i]$ we denote the $(i+1)$-st character of $S$, for $0 \leq i < m$. Likewise, by $S[i \,..\, j]$ we denote the substring of $S$ contained between the $(i+1)$-st and the $(j+1)$-st characters of $S$, for $0 \leq i \leq j < m$.

A Finite State Automaton is a quintuple $\mathcal{A} = (Q, p_0, F, \Sigma, \delta)$, where $Q$ is the set of states of the automaton, $p_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of accepting states, $\Sigma$ is the alphabet of characters labeling transitions, and $\delta$ is a partial function from $Q \times \Sigma$ to $Q$, called the transition function. If $\delta(p, c)$ is not defined for a state $p \in Q$ and a character $c \in \Sigma$, we say that $\delta(p, c)$ is an empty transition and write $\delta(p, c) =\perp$. Moreover, for all $c \in \Sigma$ we put $\delta(\perp, c) = \perp$.

In contrast with what can be observed in random texts with a uniform character distribution, it turns out that some naturally occurring phenomena in natural language texts obey a *power-law* distribution.

**Zipf's law** (cf.[Z32]), named after the Harvard linguistic professor George Kingsley Zipf (1902-1950), is one of the most interesting applications of the power-law to natural languages. In particular, Zipf's law connects the rank of a word in a natural language text with its relative frequency on the text itself as follows: given a text $T$, Zipf's law states that, with a very good approximation, the relative frequency of a word is inversely proportional to its rank. More formally, if $R$ is the number of different words in $T$, then the relative frequency $f(r)$ of a word with rank $r$ in $T$ is approximated by expression **(1)** shown below:

$$\textbf{(1)} \quad f(r) \simeq \frac{1}{r \ln(1.78R)}, \qquad\qquad \textbf{(2)} \quad f(r) \simeq \frac{(R-i+1)^k}{\sum_{j=1}^{R} j^k}.$$

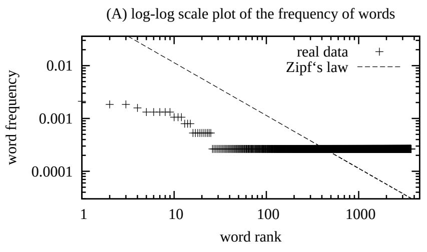
Figure 1 presents the relative frequencies of words in a random text buffer, with a uniform distribution of characters (Figure 1.A) and in a natural language text "Hamlet" (Figure 1.B) and their approximations with the Zipf's law. In contrast with the natural language text, we can observe that the relative frequency of words in the random text does not follow a Zipf's law.

Recently, in [CF03] a similar characterization has been reported for the frequencies of characters in natural language texts. Such distribution model gives a very good approximation of the relative frequency function of characters in terms of their rank both in natural language dictionaries and texts. The model is based on the following **inverse-rank power-law** of degree $k$, which states that if $R$ is the number of different characters in $T$, then the relative frequency $f(r)$ of the character with rank $r$ in $T$ is approximated by the above expression **(2)**, for a degree $k \in \mathbb{R}$ whose value is to be determined experimentally (usually $k$ ranges in the closed interval $[3 \,..\, 10]$).

## 2 The Finite State Model

The model adopted in this note is a Deterministic Probabilistic Finite Automaton (DPFA) [Paz71,VT+05], called *Extended q-Gram Model* [NMW99], which inherits the statistical structure of the string used for its construction (see below). The $q$-Grams automata are equivalent to a class of DPFA known as *stochastic k-testable automata* [GV90].

The DPFAs are models which are generative in nature. This is in contrast with the standard definition of automata in the conventional (non-probabilistic)



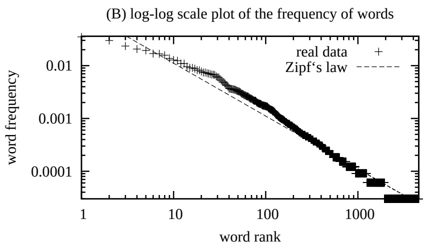


**Fig. 1.** The relative frequencies of words in a random text buffer (A) and in a natural language text (B), and their approximations with Zipf's law. The text buffer in (A) has been generated randomly with a uniform distribution of characters ($n = 200000, \sigma = 50$). The natural language text in (B) derives from the English drama "Hamlet" ($n = 174073, \sigma = 65$).

formal language theory, where strings are generated by grammars, whereas the automata are the accepting devices. Thus, if $S$ is a natural language string, then a DPFA for $S$ can be used to generate random texts which maintain the peculiar characteristics of $S$ itself. The $q$-Gram Model for a string $S$ is constructed by extracting all $q$-grams of the string $S$, for some fixed $q > 0$, and by carrying the statistical relationships between overlapping $q$-grams.

To begin with, let $S$ be a string and let $\Sigma$ be its alphabet. Given a positive integer $q$, with $q \leq lenght(S)$, we define a $q$-gram of $S$ as a substring $w$ of $S$ of length $q$, i.e., $w = S[i \,..\, i + q - 1]$, for some $0 \leq i \leq |S| - q$. We denote with $G_q(S)$ the set of all $q$-grams of $S$. We define also an occurrence function $\rho_S$ which associates to each nonempty string $w$ over $\Sigma$ the number of its occurrences in $S$, namely:

$$\rho_S(w) = |\{i \; : \; 0 \leq i \leq n - |w| \text{ and } S[i \,..\, i + |w| - 1] = w\}| \, .^1$$

A standard natural language text can be seen as a sequence of words separated by special symbols such as punctation marks, numbers, *blanks*, etc. Thus, we assume that there is a distinguished set $\Sigma_{sep} \subseteq \Sigma$ containing such symbols, used as separators between different words.

Given a value $q$, with $0 < q < n$, the $q$-Gram Automaton ($q$-GA for short) for the string $S$ is formally defined as the probabilistic finite state automaton $\mathcal{A} = (Q, p_0, F, \Sigma, \delta, \varphi)$, where

1. $Q$ is the set of all $q$-grams of $S$, i.e., $Q = G_q(S)$;
2. $p_0$ is the initial state, defined as the first $q$-gram of $S$, i.e. $p_0 = S[0 \,..\, q - 1]$;
3. $F$ is the set of final states, defined as $F = \{w \in Q \mid w[q - 1] \in \Sigma_{sep}\}$;
4. $\delta$ is the transition function defined, for each $w \in \; G_q(S)$ and $a \in \Sigma$, by

$$\delta(w, a) = \begin{cases} w[1 \,..\, q - 1].a & \text{if } w[1 \,..\, q - 1].a \in G_q(S) \\ \perp & \text{otherwise,} \end{cases}$$

[1] Notice that $0 < \rho_S(w) \leq n - q + 1$, for each $q$-gram $w$ of $S$.

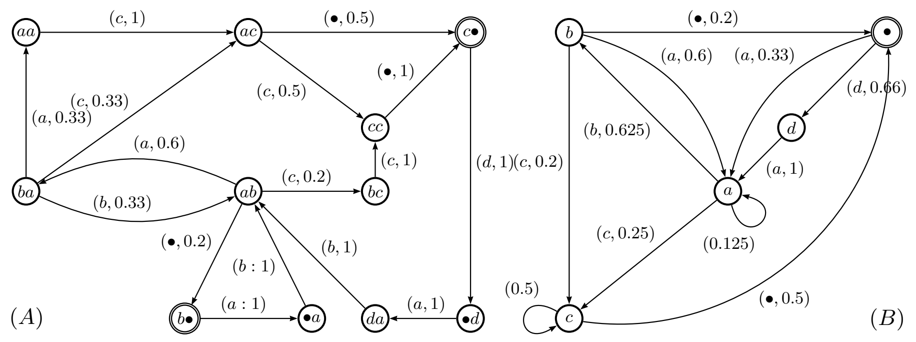


**Fig. 2.** (A) The 2-GA for the string $S = abaac \bullet dabab \bullet abacc \bullet dabcc$, where $\Sigma = \{a, b, c, d, \bullet\}$ and $\Sigma_{sep} = \{\bullet\}$. Each transition $\delta(w, c)$ of the automaton is labeled with the pair $(c, \varphi(w, c))$, where $c$ is the character which performs the transition and $\varphi(w, c)$ is the relative frequency of the transition. (B) The 1-GA for the string $S = abaac \bullet dabab \bullet abacc \bullet dabcc$.

5. $\varphi : (Q \times \Sigma) \to \mathbb{R}$ is a map which associates to each transition of the automaton its relative frequency. The map $\varphi$ is formally defined by

$$\varphi(w, a) = \begin{cases} \dfrac{\rho_S(w.a)}{\sum_{c \in \Sigma} \rho_S(w.c)} & \text{if } \delta(w, a) \neq \bot \\ 0 & \text{otherwise,} \end{cases}$$

   for each $w \in G_q(S)$ and $a \in \Sigma$.

See Figure 2 for a pictorial illustration. The construction of the $q$-GA for a string $S$ of length $n$ takes $\mathcal{O}(n + |\Sigma|^{q+1})$-time and requires $\mathcal{O}(|\Sigma|^{q+1})$-space, where $\Sigma$ is the alphabet of the string $S$.

## 3 Text Generation

In this section we present a simple algorithm for generating random texts by means of the finite state model described above. Then we present experimental evidence that the random texts so generated obey Zipf's law and the inverse-rank power law, namely they enjoy the structural and statistical characteristics of natural language texts.

```
Random-Transition($\mathcal{A}, \Sigma, w$)
1.    $r \leftarrow random(0, 1]$
2.    $\pi \leftarrow 0$
3.    $j \leftarrow 1$
4.    while $\pi < r$ do
5.        $\pi \leftarrow \pi + \varphi(w, c_j)$
6.        $j \leftarrow j + 1$
7.    return $c_j$
```

```
Generator($S, \mathcal{A}, q, \Sigma, n$)
1.    $w \leftarrow S[0 .. q-1]$
2.    $T[0 .. q-1] \leftarrow w$
3.    $i \leftarrow q$
4.    while $i < n$ do
5.        $c \leftarrow$ Random-Transition($\mathcal{A}, \Sigma, w$)
6.        $T[i] \leftarrow c$
7.        $w \leftarrow \delta(w, c)$
8.        $i \leftarrow i + 1$
9.    return $T$
```

The algorithm Generator given above takes as input a string $S$, the $q$-GA $\mathcal{A}$ for the string $S$, a dimension $q$, the alphabet $\Sigma$, and the length $n$ of the output text buffer $T$. We assume that the alphabet $\Sigma$ is an ordered finite alphabet, so that we can write $\Sigma = \{c_1, c_2, \dots, c_\sigma\}$, with $|\Sigma| = \sigma$. The algorithm starts its transitions from the initial state $w = S[0 \,..\, q-1]$. Thus the first $q$ characters of the output text $T$ will be equal to the first $q$ characters of $S$. Then it performs a loop until $n$ characters have been inserted in $T$. More precisely, at each iteration, it uses the last $q$-gram of $T$, $w$, to compute the subsequent character $c$ to be inserted. In particular $c$ is selected randomly among all possible transitions $\delta(w, c)$ in the $q$-GA, according to their frequencies $\varphi(w, c)$ (procedure Random-Transition). Then $w$ is updated to $\delta(w, c)$. Clearly the Generator algorithm takes $\mathcal{O}(n)$-time for computing a text buffer of length $n$.

The following table lists some files containing random texts (output files), all of dimension 2Mb, generated by the algorithm described above from natural language texts (source files) of different sizes and languages, with grams dimension $q = 2$. Each file is accessible via a URL of the form

http://www.ippari.unict.it/faro/fsmnlp08/file_name.txt.

| **text** | **language** | **source file** | **output file** |
|---|---|---|---|
| Hamlet (W. Shakespeare) | English | ham.txt (176Kb) | hamg3.txt |
| La Divina Commedia (D. Aligheri) | Italian | div.txt (549Kb) | divg3.txt |
| De la Terre à la Lune (J. Verne) | French | terlun.txt (335Kb) | terlung3.txt |
| Don Quijote (M. Cervantes) | Spanish | quijote.txt (2, 04Mb) | quijoteg3.txt |
| English dictionary (151.160 entries) | English | endict.txt (1, 47Mb) | endictg3.txt |
| Words beginning with $w$ (328 words) | English | wwords.txt (2, 24Kb) | wwordsg3.txt |
| Italian dictionary (277.313 entries) | Italian | itdict.txt (3, 13Mb) | itdictg3.txt |
| World Fact Book (Canterbury Corpus) | English | world192.txt (2, 35Mb) | world192g3.txt |
| The Bible (Canterbury Corpus) | English | bible.txt (3, 85Mb) | bibleg3.txt |

Below are presented three examples of random texts, of length $n \geq 100$, generated by the algorithm described above from strings $S_1$, $S_2$, and $S_3$ with grams dimensions $q \in \{1, 2\}$.

**Example 1.** $S_1 =$"*abaac dabab abacc dabcc*" (this is the strung used in Figure 2):

(q=1) *ab daabababab ab ababcc abacc dabab dababc dabababaccccc abababababaacc dababababab dababcc ab dab dababa*

(q=2) *ababcc dab ab abacc dabcc dabac dabaac dabcc dab abaacc dabac dabac dabab abcc dab abacc dabacc dabc*

**Example 2.** $S_2 =$ concatenation of the 328 different words of length 6 of the English dictionary, beginning with the letter $w$:

(q=1) *we warif wolviter w wices whofeshs wadeddd wommmpier wilads waxeathaveshs wally war wind wis waldooup*

(q=2) *waffy wra winne wifer wiper whoolver woo wafed weekly wagong whing wincern weaker wrer woolver woren*

**Example 3.** $S_3 =$ concatenation of the 1389 different words of length 6 of the Italian dictionary, beginning with the letter $a$

(q=1) *aro acca agnfi ac ara ara alito andereatealbiannna arga aloniacci affi arma ale aluca atsiti affe*

(q=2) *abbaggia azoiolo apone amperemia abdulsa assi annomie arreso agrai amarpo aucideraspio aliata apta*

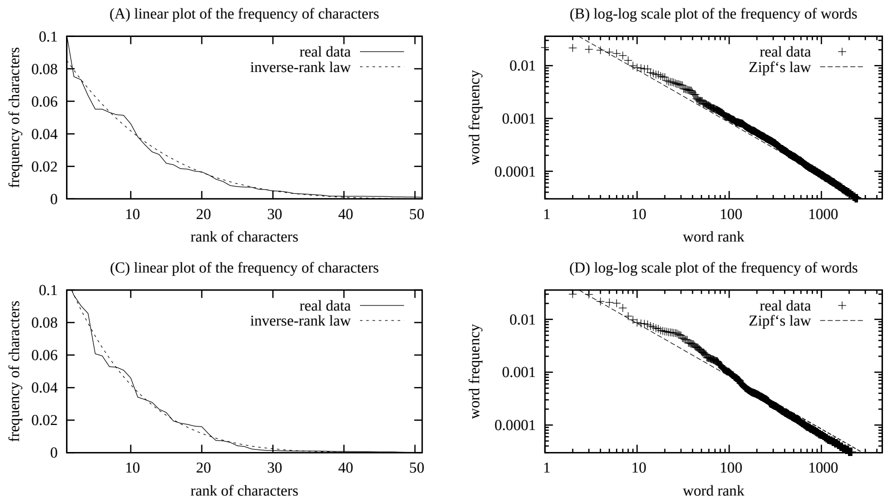


**Fig. 3.** The relative frequencies of characters (A-C) and words (B-D) in two different randomly text buffers of dimension $2MB$, generated from two 2-GA, and their approximations with the inverse-rank power law and Zipf's law, respectively. The text buffer in (A-B) comes from the English drama "Hamlet" ($n = 174073, \sigma = 65$). The text buffer in (C-D) comes from the Italian poem "La Divina Commedia" ($n = 548159, \sigma = 62$).

Observe that texts generated by the 1-GAs contain words which are not closely related with the structure of the source string. Short words like "*ab*" in Example 1, "*w*" in Example 2, and "*ac*" in Example 3, appear in the generated texts together with quite long words like "*abababababaacc*" (in Example 1), "*waxeathaveshs*" (in Example 2) and "*andereatealbiannna*" (in Example 3). Moreover, strings containing long sequences of a same character, like "*dababaccccc*" (in Example 1) and "*wommmpier*" (in Example 2) can occur. Instead, the length of the words generated by the 2-GAs are very close to the length of the words in the source string and anomalies due to character repetitions are not present.

Figure 3 shows the relative frequencies of characters and words in two different text buffers of dimension $2Mb$, generated from 2-GAs for two different natural language texts. Observe that for both text buffers the relative frequencies of characters are well approximated by an inverse-rank power law, of degree 4.7 and 5.9, while the relative frequency of words follows closely a Zipf's law.

## 4 Conclusion and Plans for Future Works

In this note we have presented a preliminary study of a finite state model for generating random text buffers with the same structure of natural language texts. The model can be used to generate large corpora of data for testing text processing algorithms for data-compression and pattern-matching. We intend

| (A) words | Italian | English | German |
|---|---|---|---|
| *airships* | 0.0279 | **0.1323** | 0.0576 |
| *beautiful* | 0.0989 | **0.1355** | 0.1022 |
| *cetrioli* | **0.1391** | 0.1000 | 0.0610 |
| *crazed* | 0.0362 | **0.2346** | 0.0378 |
| *cucumbers* | 0.0821 | **0.1871** | – |
| *dirigibili* | **0.2186** | 0.1360 | 0.0949 |
| *equazioni* | **0.4025** | – | – |
| *hydrophily* | – | **0.3123** | – |
| *idrofilo* | **0.1550** | 0.0963 | 0.0536 |
| *impazzivo* | **0.1947** | – | 0.0691 |
| *inno* | **0.4825** | 0.1958 | 0.1793 |
| *meravigliosi* | **0.2221** | 0.0965 | 0.1035 |
| *none* | 0.2018 | **0.2063** | 0.1039 |
| *piuolo* | **0.1296** | – | – |

| (B) words | Italian | English | German |
|---|---|---|---|
| *quadrato* | **0.3313** | 0.2491 | 0.2223 |
| *quaglia* | **0.4088** | 0.2565 | – |
| *quando* | **0.4598** | 0.2687 | 0.2894 |
| *quantizzammo* | **0.4086** | 0.1884 | 0.1915 |
| *raggi* | **0.2519** | 0.1027 | 0.0639 |
| *spokes* | 0.0938 | **0.2067** | 0.1360 |
| *spring* | 0.0851 | **0.2559** | 0.1900 |
| *stare* | **0.1795** | 0.0903 | 0.1377 |
| *stars* | 0.1002 | **0.2112** | 0.1209 |
| *state* | **0.2354** | 0.1738 | 0.1564 |
| *testamenti* | **0.2631** | 0.1519 | 0.1451 |
| *trentesimo* | **0.2327** | 0.1168 | 0.1483 |
| *why* | – | **0.1159** | – |
| *wills* | 0.0994 | **0.2656** | 0.1713 |

| (C) sentences | italian | english | french |
|---|---|---|---|
| *A buon cavallo non manca sella* | **0.2189** | – | – |
| *A buon intenditor poche parole* | **0.2275** | – | – |
| *A picture is worth a thousand words* | – | **0.2132** | – |
| *A tavola non si invecchia* | **0.2630** | – | – |
| *A word to the wise is sufficient* | – | **0.2409** | – |
| *Action speak louder than words* | – | **0.2276** | – |
| *Buon sangue non mente* | **0.2492** | – | – |
| *Chi domanda cio che non dovrebbe, ode cio che non vorrebbe* | **0.2755** | – | – |
| *Chi dorme d'agosto, dorme a suo costo* | **0.1837** | – | – |
| *Hard words break no words* | – | **0.1870** | – |
| *Il faut tourner sa langue sept fois dans sa bouche avant de parle* | – | – | **0.2604** |
| *In the land of the blind, the one eyed man is king* | – | **0.3012** | – |
| *La parole est d'argent, mais le silence est d'or* | – | – | **0.2185** |
| *Le parole sono femmine e i fatti sono maschi* | **0.1822** | – | – |
| *Nel mezzo del cammin di nostra vita* | **0.2672** | – | – |
| *The cat will mew and dog will have its day* | – | **0.2882** | – |
| *To be or not to be, that is the problem* | – | **0.3282** | – |

**Fig. 4. (A-B)** The similarity coefficient values for a set of words over the Italian, English and German dictionaries. The symbol "–" indicates that the word is not recognized by the automaton. **(C)** The similarity coefficient values for a set of English, Italian and French proverbs. The sentences have been tested with three 2-GAs constructed over three different natural language texts: the English drama "Hamlet" by William Shakespeare; the Italian poem "La Divina Commedia" by Dante Alighieri; the French novel "De la Terre à la Lune" by Jules Verne.

to investigate further applications of the Extended $q$-Gram Model in automatic music generation, for creating original music pieces from input scores, and in the field of image precessing for automatic texture generation.

Another major application field which we intend to investigate relates to the *Language Identification* problem, which has applications in many areas such as in spelling and grammar correction, in database search engine, etc. For instance, given a $q$-GA $\mathcal{A} = (Q, p_0, F, \Sigma, \delta, \varphi)$ and an input string $w$ of length $m$, we can associate to $w$ a similarity coefficient value, $\Gamma_{\mathcal{A}}(w)$, computed as the average relative frequency of transitions $(w[i\,..\,i+q-1], w[q])$, for $0 \leq i < |w| - q$. A similar application, based on Markov Models, has been presented in [D94].

Then the language identification problem can be addressed by parsing the input string with different $q$-GAs constructed over different natural language texts and taking the language which leads to the higher similarity coefficient value.

Figure 4(A-B) presents the similarity coefficient values obtained by parsing a set of English and Italian words with 2-GAs constructed over three different natural language dictionaries whereas Figure 4(C) presents the similarity coefficient values for a set of English, Italian and French proverbs.

## References

[AJB99] Albert, R., Jeong, H., and Barabási, A.-L. (1999). Internet: Diameter of the World Wide Web. *Nature*, 401:130–131.

[ACR99] Allauzen, C., Crochemore, M., and Ranot., M. (1999). Factor oracle: a new structure for pattern matching. *Theory and Practice of Informatics*, number 1725, pages 291-306, Milovy, Czech Republic.

[CF03] Cantone, D. and Faro, S. (2003). On the frequency of characters in natural language texts. *Proc. of the* $3^{rd}$ *AMAST Workshop on Language Processing, AMILP 2003*, pp. 10–24.

[CF04] Cantone, D. and Faro, S. (2003). Fast-Search Algorithms: New Efficient Variants of the Boyer-Moore Pattern Matching Algorithm. *Proc. Lecture Notes in Computer Science, Springer Berlin*, Volume 2647 pp. 622.

[D03] Deorowicz, S. (2003). Silesia Corpus *Silesian University of Technology, Poland*, http://data-compression.info/Corpora/SilesiaCorpus .

[D94] Dunning, T. (1994). Statistical identification of language. *Technical Report CRL MCCS-94-273,* Computing Research Lab, New Mexico State University.

[GV90] García, P., Vidal, E. (1990). *Inference of K-testable Languages in the Strict Sense and Applications to Syntactic Pattern Recognition.* IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 12, Issue 9, pp. 920-925.

[NMW97] Ney, H., Martin, S., and Wessel, F. (1997). *Corpus-Based Statiscal Methods in Speech and Language Processing*, S. Young and G. Bloothooft, Kluwer Academic Publishers, ch. Statistical Language Modeling Using Leaving-One-Out, pp. 174–207.

[NMW99] Nevill-Manning, C. G. and Witten, I. H. (1999). Protein is Incompressible *Data Compression Conference*, pp. 257–266, http://data-compression.info/Corpora/ProteinCorpus .

[Paz71] Paz, A. (1971). Introduction to Probabilistic Automata. *Academic Press, New York, NY.*

[RB97] Ross, A. and Bell, T. (1997). The Canterbury Corpus *University of Canterbury, New Zeland*, http://corspus.canterbury.ac.nz .

[VT+05] Vidal, E., Thollard, F., De La Higuera, C., Casacuberta, F., Carrasco, R.C. (2005). *Probabilistic Finite State Automata – Part I and II.* IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 27, Issue 7, July 2005, pp. 1013-1039.

[Z32] Zipf, G. K. (1932). *Selective Studies and the Principle of Relative Frequency in Language*, volume 23. Harvard University Press, Cambridge, MA.