CHAPTER V

MARKOV MODEL APPROACH OF LANGUAGE MODELS

5.1 INTRODUCTION

Over a few decades, speech recognition and language understanding are two major research thrusts that have traditionally been approached as problems in linguistics and acoustic phonetics, where a range of acoustic phonetic knowledge has been brought to bear on the problem with remarkably little success (Rabiner, 1989). This chapter focuses about the statistical methods for language, language models and a synthesis of the theoretical fundamentals of language processing through a well-defined mathematical and statistical formalism.

Language is the knowledge of a creative communication system composed of meaningful elements, which can be combined in many ways to produce sentences. Thus, language has been viewed as a vessel for thought and as a reflection of thought, but not as something that determines thought. Despite small differences in the use of language by different people who share a language, and despite the larger differences among the various languages of the world in their structure and vocabularies, there may be some universal features of all human languages. If language is conceived as a set of rules by which an infinite number of sentences can be generated, using a stock of words
that constantly expands to cover all concepts one may choose to express, then humans are the only creatures yet known to have a command of language.

Human language is highly ambiguous at various levels like acoustic, morphological, syntactic and semantic. The primary feature of human language is the duality of patterning, where it enables to use language in a very economic way for a virtually infinite production of linguistic units. The two types of models that describe language are acoustics and statistical language models. Both acoustic modeling and language modeling are important parts of modern statistically based speech recognition algorithms.

In general, the acoustic modeling (Young, 1996) defines in training a set of acoustic models for the words or sounds of the language by establishing the statistical relationship between acoustic features $X$, for each word or sound and a word sequence $W$, whereas, the language modeling (Jelinek, 1997; Rosenfeld, 1996) describes the probability of a sequence of words that form a valid sentence in the task language.

A language model is an important component of a modern speech recognition system, where it assigns probabilities to strings of symbols and the fundamental approach to large vocabulary continuous speech recognition [73] is to assume a simple probabilistic model of speech production. Let $P(W|Y)$ be the probability for a specified sequence $W$, given an acoustic observation sequence, and hence it is to determine a word string $\hat{W}$ satisfying the decoded
string has the Maximum a posteriori Probability (MAP) \( \hat{w} = P(W|Y) = \max_{w} P(W|Y) \). Using Baye’s rule, this expression can be written as \( P(W|Y) = \frac{P(Y|W)P(W)}{P(Y)} \), where the terms \( P(W|Y) \) and \( P(W) \) are called the acoustic model (estimates the probability of a sequence of acoustic observations on the considered word string) and language models (describes the probability associated with a postulated sequence of words) respectively.

### 5.2 Statistical Language Modeling (SLM)

Statistical language modeling (SLM) techniques were first applied in speech recognition, where the goal of SLM is to predict the next term given the terms previously uttered. The adaptation of SLM to ad hoc document retrieval (proposed by Ponte and Croft, 1998; Hiemstra and Kraaij, 1999) is typically referred to as the language modeling (LM) approach. Since then LM has become a widely accepted, effective, and intuitive retrieval model with many variant realizations, Berger and Lafferty (1999) proposed an overview of Information Retrieval as Statistical translation.

Language models are attractive because of their foundations in statistical theory, the great deal of complementary work on language modeling in speech recognition and natural language processing, and the fact that very simple language modeling retrieval methods have performed quite well empirically (Zhai and Lafferty, 2002).
Figure 5.1 describes the Call Processing Language (CPL), which is used to describe and control Internet telephony services. The purpose of CPL is to be powerful enough to describe a large number of services and features, but it is limited in power so that it can run safely on Internet telephony servers. An example of one modeled service is shown in Figure 5.1.

The intention of the CPL language is to make it impossible for users to do anything more complex than describing Internet telephony services. The structure of the language maps closely to its behavior, so any CPL service developer can easily understand and validate the script even from these graphical models.
Statistical language modeling (SLM) is the attempt to capture regularities of natural language for the purpose of improving the performance of various natural language applications. By and large, statistical language modeling amounts to estimating the probability distribution of various linguistic units, such as words, sentences, and whole documents. Statistical language modeling is crucial for a large variety of language technology applications. These include speech recognition (the origin of SLM), machine translation, document classification and routing, optical character recognition, information retrieval, handwriting recognition, spelling correction, and many more.

5.3 PRELIMINARIES AND NOTATIONS

This section provides necessary preliminaries and notations related to the proposed work. Some are discussed below.

**Language Model**

The probability $p(W)$ of a sequence of words $W = w_1, w_2, \ldots, w_L$ is computed by a language model (LM). Then, $p(W)$ can be expressed as follows:

$$p(W) = p(w_1, \ldots, w_n) = \prod_{i=1}^{n} p(w_i \mid w_{0}, \ldots, w_{i-1})$$  \hspace{1cm} (5.1)

In general, the language model can be expressed and it is impossible to reliably estimate the conditional word probabilities $P(w_n \mid w_1, w_2, \ldots, w_{n-1})$ for all words, and hence it is convenient to use an N-gram word model, using approximation method.
Language model evaluation

The language model evaluation can be performed in several ways such as,

- Random sentence generation;
- Words reordering in sentences;
- Perplexity; and
- Integration in an existing speech recognition system.

Among them, the actual performance of language models to assess the quality of a given language modeling technique is often reported in terms of perplexity, which is the most used measurement for language model evaluation.

N-gram Language Model

Since the language model tries to estimate the probability of word sequence defined in (5.1), and using Markovian assumption, the history of words can be reduced to the last n-1 words and is shown below,

\[
P(w_i | w_1 w_2 \ldots w_{i-1}) \approx P(w_i | w_{i-n+1} \ldots w_{i-1})
\]

The most widely used language models are n-gram language models, where it estimates the probability of the next word based on the last n-1 preceding words.

For \( n = 1 \ldots 3 \), then we have

- Unigram language model (n=1);
- Bigram language model (n=2);
- Trigram language model (n=3).
Unigram language model

The unigram language model considers all words to be independent. It means that no previous history of words is required, which is

\[ p(w_k \mid w_{i}^{k-1}) \approx p(w_k) \]  \hspace{1cm} \ldots (5.3)

Thus, the probability estimation for the unigram model will be,

\[ p(w^n_p) = \prod_{k=1}^{n} p(w_k) \]  \hspace{1cm} \ldots (5.4)

Bigram language model

In bigram model, the estimation of probability of a current word \( w_i \) only depends on the identity of the immediately preceding word, denoted as

\[ p(w_k \mid w_{i}^{k-1}) \approx p(w_k \mid w_{k-1}) \]  \hspace{1cm} \ldots (5.5)

Thus, the probability estimation formula for bigram language model will be,

\[ p(w^n_p) = p(w_1) \prod_{k=2}^{n} p(w_k \mid w_{k-1}) \]  \hspace{1cm} \ldots (5.6)

Trigram language model

In trigram model, instead of conditioning the probability of a word on the identity of just preceding word, it is to condition the probability on the identity of the preceding n-1 and n-2 words respectively, then the model will be

\[ p(w_k \mid w_{i}^{k-1}) \approx p(w_k \mid w_{k-1}, w_{k-2}) = p(w_k \mid w_{k-2}^{k-1}) \]  \hspace{1cm} \ldots (5.7)
Thus, the probability estimation formula for trigram language model will be,

\[
p(w_n^w) = p(w_1).p(w_2 | w_1). \prod_{k=3}^{n} p(w_k | w_{k-1}, w_{k-2})
\]  

\ldots (5.8)

5.4 SMOOTHING AND PROBABILITY ESTIMATION

The basic idea behind the language modeling approach is to estimate a model for each document, and then rank documents by the likelihood of the query according to the estimated language model. A core problem to address language modeling estimation is smoothing, which adjusts the maximum likelihood estimator so as to correct the inaccuracy due to data sparseness. The primitive objective of smoothing is to assign a non-zero probability to the unseen words and to improve the accuracy of word probability estimation and many smoothing methods have been proposed, mostly in the context of speech recognition tasks (Chen and Goodman, 1998). In general, all smoothing methods attempt to discount the probabilities of the words seen in the text, and to then assign the extra probability mass to unseen words according to the language model [97].

Let us consider a speech recognition task for illustration. In this process, the objective is to find the sentence s that maximizes \( p(A | s) \cdot p(s) \) for a given acoustic signal A. If \( p(s) \) is zero, then \( p(A | s) \cdot p(s) \) will be zero and the string s will never be considered regardless of the acoustic signal A. Thus, whenever a string s such that \( p(s) = 0 \) occurs during a speech recognition task, an error will be appeared. Assigning all strings a non-zero probability helps prevent
errors in speech recognition. Hence, smoothing is used to address this problem.

One simple smoothing technique is to pretend each bigram occurs once more than it actually does (Lidstone, 1920; Jhonson, 1932; Jeffreys, 1948), yielding

\[
p_{s+1}(w_j \mid w_{j-1}) = \frac{C(w_{j-1}, w_j) + 1}{\sum_w [C(w_{j-1}, w_j) + 1]} = \frac{C(w_{j-1}, w_j) + 1}{\sum_w C(w_{j-1}, w_j) + |v|} \quad \ldots (5.9)
\]

Where,

v is the vocabulary, the set of all words being considered,

P(w_i \mid w_{i-1}) be the frequency with which the word w_i occurs given the last word is w_{i-1} and

C(w_{i-1} w_i) denotes the number of times in the bigrams w_{i-1} w_i occurs in the given text.

Let us assume vocabulary v to be the set of all words occurring in the training data set S, so that |v| = 11 (if v is to be infinite, then denominator is also infinite and all probabilities are set to zero, vocabularies are typically fixed to be tens of thousands of words).

Consider an example,

For the sentence *Tom wrote a comic*, then

\[
P(Tom \text{ wrote a comic}) = p(Tom \mid w_{bos}) \ p(\text{wrote} \mid Tom) \ p(a \mid \text{wrote}) \ p(\text{comic} \mid a) \ p(w_{eos} \mid \text{comic})
\]

\[
= \frac{2}{14} \times \frac{2}{12} \times \frac{3}{14} \times \frac{2}{13} \times \frac{2}{13} \approx 0.0001
\]
It is seen that the estimation of a sentence *Tom wrote a comic* occurs about once every ten thousand sentences, which is more reasonable than the zero probability assigned by the maximum likelihood model. While smoothing is a central issue in language modeling, many methods have been proposed in the literature, mostly in the context of speech recognition tasks. Previous studies (Nadas Arthur, 1984; Katz, 1987; Church and Gale, 1991) compared a small number of existing methods on a single corpus and using a single training data size. Among them, the most widely used relatively efficient smoothing methods are such as Jelinek and Mercer (1980), Katz (1987), Church and Gale (1991), Good, I.J (1953). The description of the methods is as follows:

**5.4.1 JELINEK – MERCER SMOOTHING METHOD**

The Jelinek-Mercer smoothing method (*Jelinek and Mercer, 1980*), sometimes referred to as linear interpolation or mixture model, involves a linear interpolation of the maximum likelihood model $p(t|d)$ with the collection model $p(t)$, using a coefficient $\lambda$ to control the influence of each model

$$P(t|\theta_d) = (1-\lambda) \cdot p(t|d) + \lambda \cdot p(t) \quad \ldots \text{(5.10)}$$

The probability of a term in the collection model is defined by:

$$p(t) = \frac{\sum_d n(t,d)}{\sum_d n(d)} \quad \ldots \text{(5.11)}$$

This form of smoothing was derived from a linguistic perspective by Hiemstra (1999) and from a formal basis using the Hidden Markov Model by Miller *et al.* (1999). On the surface, the use of language models appears fundamentally different from vector space models with TF-IDF weighting schemes, however
Zhai and Lafferty (2002) pointed out an interesting connection between the language modeling approach and the heuristics used in the traditional models. The use of the collection model $p(t)$ as a reference model for smoothing document language models implies a retrieval formula that implements TF-IDF weighting heuristics and document length normalization (Hiemstra and Kraaij, 1999). Representing the document model as a mixture between the document and the collection is the most popular type of language model, and is usually referred as the standard language modeling approach (Azzopardi, et al. 2005).

5.4.2 GOOD – TURING SMOOTHING METHOD

The Good-Turing estimate method (Good, 1953) is central to many smoothing techniques. It states that for any n-gram that occurs $r$ times, one should pretend that it occurs $r^*$ times, where

$$r^* = (r + 1) \frac{n_{r+1}}{n_r} \quad \ldots (5.12)$$

and where $n_r$ is the number of grams that occur exactly $r$ times in the training data. While converting this counts to a probability measure, then normalization is possible such that for n-gram $\alpha$ with $r$ counts, then

$$PGT(\alpha) = \frac{r^*}{N} \quad \ldots (5.13)$$

where $N$ is the total number of counts in the distribution.

Finally, the Good-Turing estimate yields absurd values when $n_r = 0$, and it is necessary to smooth the values of $n_r$, i.e., to adjust $n_r$ so that they are all above zero. Gale and Sampson (1995) have proposed a simple and efficient algorithm
for smoothing these values. Good concludes that the Good–Turing estimate is only used for small counts \(r \leq k\), and \(n_r\) is generally fairly high for the values of \(r\). In practice, the Good-Turing estimate is not used by itself for n-gram smoothing, because it does not include the interpolation for higher order models and however it is used as a tool in several smoothing techniques.

**5.4.3 KATZ SMOOTHING METHOD**

The further smoothing technique besides Jelinek-Mercer smoothing used widely in speech recognition is due to Katz (1987). Katz smoothing extends the intuitions of Good-Turing by adding the interpolation of higher order models with lower-order models.

Let us consider Katz smoothing for bigram models. It states that for every count \(r>0\), a discount ratio \(d_r\) is calculated, and any bigram with \(r>0\) counts is assigned a corrected count of \(d_r \cdot r\) counts. Then to calculate a given conditional distribution \(p(w_i|w_{i-1})\), the non-zero counts are discounted according to \(d_r\), and the counts subtracted from the non-zero counts in that distribution are assigned to the bigrams with zero counts. These counts assigned to the zero-count bigrams are distributed proportionally to the next lower-order n-gram model i.e, the unigram model. If the original count of a bigram \(C(w_{i-1}^i) = r\), then its corrected count is defined as,

\[
C_{\text{katz}}(w_{i-1}^i) = \begin{cases} 
  d_r \cdot r, & \text{if } r > 0 \\
  \alpha \cdot p_{\text{katz}}(w_i), & \text{if } r = 0
\end{cases}
\]  

\[\ldots (5.14)\]
where $\alpha$ is chosen such that the total number of counts in the distribution $\sum w_i C_{katz}(w_i)$ is unchanged. To calculate one such a probability from the corrected count, the normalization is done using an expression,

$$P_{katz}(w_i | w_{i-1}) = \frac{C_{katz}(w_i)}{\sum w_i C_{katz}(w_i)} \quad \ldots (5.15)$$

The values of $d_r$ is chosen in such a way that the resulting discounts are proportional to the discounts predicted by the Good-Turing estimate, and such that the total number of counts discounted in the global bigram distribution is equal to the total number of counts that should be assigned to bigrams with zero counts according to the Good-Turing estimate. Katz smoothing for higher order n-gram models is defined analogously, the bigram model is defined in terms of the unigram model and in general, katz n-gram model is defined in terms of the Katz (n-1) gram model, similar to Jelinek-Mercer smoothing method. While performing the recursion or iteration process, the Katz unigram model is taken to be the Maximum Likelihood unigram model which is shown as,

$$P_{katz}(w_i) = P_{ML}(w_i) = \frac{C(w_i)}{\sum w_j C(w_j)} \quad \ldots (5.16)$$

Katz concludes that his algorithm performs atleast as well as Jelinek-Mercer smoothing and Nadas smoothing (Nadas, 1984) using 750, 000 words of training data from an office correspondence database.
5.4.4 CHURCH – GALE SMOOTHING METHOD

Church and Gale (1991) described a smoothing method which combines Good-Turing and Katz estimates with a method for merging lower and higher order models. To describe this method, consider the Good-Turing estimate directly for each bigram with \( r \) counts, the corrected count is assigned using an expression:

\[
\hat{r}^* = (r + 1) \frac{n_{r+1}}{n_r}.
\]

It leads the undesirable effect of giving all bigrams with zero count and therefore the corrected count assigned by an interpolative model to a bigram \( w_{i-1} \) with zero counts is considered. Let the bigram model with zero counts be defined as,

\[
P(w_i | w_{i-1}) \alpha P(w_i) \quad \ldots (5.17)
\]

To convert this probability to a count, it is to multiply the total number of counts in the distribution. The expression is shown as,

\[
P(w_i | w_{i-1}) \sum_{w_i} C(w_{i-1}) \alpha P(w_i) \sum_{w_i} C(w_{i-1}) = P(w_i) C(w_{i-1}) \alpha P(w_i) P(w_{i-1}) \quad \ldots (5.18)
\]

Thus, \( P(w_{i-1}) \cdot P(w_i) \) may be a good indicator of the corrected count of a bigram \( w_{i-1} \) with zero counts. In this method, bigrams \( w_{i-1} \) are partitioned according to the value \( P_{ML}(w_{i-1}) \). \( P_{ML}(w_i) \). That is, it divides the range of possible \( P_{ML}(w_{i-1}) \). \( P_{ML}(w_i) \) values into a number of partitions, and all bigrams associated with the same sub range are considered to be in the same bucket. Then, each bucket is treated as a distinct probability distribution and Good-Turing estimation is performed within each. For a bigram in bucket \( b \) with \( r_b \) counts, the corrected counts is calculated using an expression,
\[ r_b^* = (r_b + 1) \cdot \frac{n_{b,r+1}}{n_{b,r}} \] \ldots (5.19)

where the counts \( n_{b,r} \) include only those bigrams within bucket \( b \). Church and Gale concludes that his algorithms partitioned the range of possible \( P_{ML}(w_{i-1}) \) \( P_{ML}(w_i) \) values into about 35 buckets, with three buckets in each factor of 10. To smooth the value of \( n_{b,r} \) for the estimates, the methods proposed by Good (1953) and Katz (1987) were used.

### 5.5 EXPERIMENTAL SETUP

In general, many research works are more related with linguistics and corpora databases for language modeling (such as words, word sequences, and syntactically related pairs of words) based on smoothing methods [16][17][87]. Apart from those databases, this section focused an illustration solely related to the context of bioinformatics setup. Among several smoothing methods, Good-Turing methods provide a simple estimate of the total probability of the observed objects that are consistent with the total probability assigned to the unseen objects.

For illustration, let us consider the following DNA (adenine, thymine, cytosine and guanine or simply A, T, C and G) sequences:

```
AAGAGTGCAAGCCGTGCGAGATATCCAAGAGATCC
```

To implement the bigram counts (\( n=2 \)) for the above sequence, the observed counts and observed frequencies were computed initially which is shown in
Table 5.1 and Table 5.2. Using Good Turing smoothing method, the probabilities are computed based on equation 5.13. For computing Good-Turing smoothing probabilities, an R program is constructed and shown in Appendix.

Table 5.1: Observed Bigram counts of sequences

<table>
<thead>
<tr>
<th>C (X, Y)</th>
<th>A</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>G</td>
<td>6</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>T</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>TOTAL</td>
<td>12</td>
<td>10</td>
<td>8</td>
<td>5</td>
<td>35</td>
</tr>
</tbody>
</table>

Table 5.2: Observed Bigram frequencies of sequences

<table>
<thead>
<tr>
<th>O (X, Y)</th>
<th>A</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>6/35</td>
<td>0</td>
<td>3/35</td>
<td>2/35</td>
<td>11/35</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>3/35</td>
<td>3/35</td>
<td>2/35</td>
<td>8/35</td>
</tr>
<tr>
<td>T</td>
<td>3/35</td>
<td>2/35</td>
<td>0</td>
<td>0</td>
<td>5/35</td>
</tr>
<tr>
<td>TOTAL</td>
<td>12/35</td>
<td>10/35</td>
<td>8/35</td>
<td>5/35</td>
<td>1</td>
</tr>
</tbody>
</table>

Based on the number of bigram counts from the observed data, the following table summarizes the estimation of probabilities using Good-Turing smoothing method.
Table 5.3: Output of Good-Turing Probabilities of Bigram sequences using R Program

<table>
<thead>
<tr>
<th>Bigram counts of sequences</th>
<th>r</th>
<th>r+1</th>
<th>N_r</th>
<th>N_r+1</th>
<th>r*</th>
<th>P_{GT^*}(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>GA</td>
<td>6</td>
<td>7</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>AG</td>
<td>5</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>0.171</td>
</tr>
<tr>
<td>TG</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>3.75</td>
<td>0.107</td>
</tr>
<tr>
<td>GT</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>3.75</td>
<td>0.107</td>
</tr>
<tr>
<td>CG</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>AC</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>3.75</td>
<td>0.107</td>
</tr>
<tr>
<td>CC</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>GC</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TA</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>AT</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>8</td>
<td>0.229</td>
</tr>
<tr>
<td>CT</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>3.75</td>
<td>0.107</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>0.828</strong></td>
</tr>
</tbody>
</table>

From the above table, it is observed that for high values of $r$ (for high frequency bigrams), $N_{r+1}$ is quite likely to be zero. For the sequence GA, there exists high frequency bigrams as 6 and which is greater than a threshold value (assumed to be $k = 5$). Therefore for any $r>k$, $r^*$ should be replaced as $r$. Again the probabilities $P_{GT^*}(X)$ are computed and now the total probabilities approaches to one. The modified table is shown as below.
### Table 5.4: A Modified Result of Good-Turing estimates of Probabilities

<table>
<thead>
<tr>
<th>Bigram counts of sequences</th>
<th>r</th>
<th>r+1</th>
<th>N_r</th>
<th>N_r+1</th>
<th>r'</th>
<th>P_{GT}^{*}(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>GA</td>
<td>6</td>
<td>7</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>0.171</td>
</tr>
<tr>
<td>AG</td>
<td>5</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>0.171</td>
</tr>
<tr>
<td>TG</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>3.75</td>
<td>0.107</td>
</tr>
<tr>
<td>GT</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>3.75</td>
<td>0.107</td>
</tr>
<tr>
<td>CG</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>AC</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>3.75</td>
<td>0.107</td>
</tr>
<tr>
<td>CC</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>GC</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TA</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>AT</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>8</td>
<td>0.229</td>
</tr>
<tr>
<td>CT</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>3.75</td>
<td>0.107</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>35</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>0.999</strong></td>
</tr>
</tbody>
</table>

Further computations in this algorithm leads to obtain a smoothing curve fitted for a polynomial function to the observed values of \((r, N_r)\). The values of \(r\) are plotted against the values of \(N_r\) (shown in Figure 5.2) which indicates the fitness of polynomial smoothing curve with \(R^2 = 57\%\), as model accuracy.
In this chapter, a probabilistic based smoothing approach for language models was demonstrated. Among several existing smoothing techniques, this chapter imported Good-Turing smoothing technique over a DNA sequence, which acts as a vital task of the bioinformatics. For convenience, the computational task is carried out for bigram models and thereby bigram counts were observed for the DNA sequence. The corresponding probabilities $P_{GT}(X)$ were computed using R program. For the specified values of $(r, N_r)$, a polynomial smoothing curve is drawn and shown in Figure 5.2.