



Tribhuvan University
Institute of Science and Technology

Predicting Sentence Using N-Gram Language Model For Nepali Language

Dissertation

Submitted to

Central Department of Computer Science and Information Technology

Kirtipur, Kathmandu, Nepal

In partial fulfillment of the requirements

for the Master's Degree in Computer Science and Information Technology

by

Ananda K.C.

December, 2012



Tribhuvan University
Institute of Science and Technology

Predicting Sentence Using N-Gram Language Model For Nepali Language

Dissertation

Submitted to

Central Department of Computer Science and Information Technology

Kirtipur, Kathmandu, Nepal

In partial fulfillment of the requirements

for the Master's Degree in Computer Science and Information Technology

by

Ananda K.C.

December, 2012

Supervisor

Prof. Dr. Shashidhar Ram Joshi



Tribhuvan University

Institute of Science and Technology

Central Department of Computer Science and Information Technology

Student's Declaration

I hereby declare that I am the only author of this work and that no sources other than the listed here have been used in this work.

.....

Ananda K.C.

December, 2012



Tribhuvan University

Institute of Science and Technology

Central Department of Computer Science and Information Technology

Supervisor's Recommendation

We hereby recommend that this dissertation prepared under our supervision by **Mr. Ananda K.C.** entitled “**Predicting Sentence Using N-Gram Language Model For Nepali Language**” be accepted as partial fulfillment of the requirements for the degree of M. Sc. in Computer Science and Information Technology. In our best knowledge this is an original work in computer science.

.....

Prof. Dr. Shashidhar Ram Joshi

Department of Electronics and Computer Engineering

Tribhuvan University

Institute of Engineering

Pulchowk

(Supervisor)



Tribhuvan University

Institute of Science and Technology

Central Department of Computer Science and Information Technology

LETTER OF APPROVAL

We certify that we have read this dissertation and in our opinion it is satisfactory in the scope and quality as a dissertation in the partial fulfillment for the requirement of Master's Degree in Computer Science and Information Technology.

Evaluation Committee

.....
Prof. Dr. Shashidhar Ram Joshi

Department of Electronics and Computer Engineering
Tribhuvan University
Institute of Engineering
Pulchowk
(Supervisor)

.....
Assoc. Prof. Dr. Tanka Nath Dhamala
Head of Department (HOD)

Central Department of Computer Science and
Information Technology (CDCSIT)
Tribhuvan University
Kirtipur

(External Examiner)

(Internal Examiner)

ACKNOWLEDGEMENT

The journey of this research and my study would have not been possible without cooperation, warm support and company of many others.

I would like to extend my, first and foremost, gratitude and sincerest thanks to my respected Supervisor Prof. Dr. Shashidhar Ram Joshi, Department of Electronics and Computer Engineering, Institute of Engineering, Pulchowk for his impressive tutelage, constructive criticism and intellectual support bestowed for me sacrificing his invaluable time . His crucial role to make this report culminate is indescribable.

I would like to express my gratitude to the respected teachers and others staffs of Central Department of Computer Science and Information Technology (CDCSIT) for granting me broad knowledge and inspirations within the time period of two years.

I am tremendously grateful to Mr. Tej Shahi for providing the supportive environment to conduct this research and for his sharp insights, constructive comments and tireless guidance during the research and for the whole study period.

I would like to thank Dr. Keith Trnka (PhD graduate in Computer and Information Sciences from the University of Delaware), Dr. Andreas Stolcke (Principal Researcher at Conversational Systems Lab at Microsoft in Mountain View) for their continuous support and encouragement during the entire course of the dissertation.

My two years at CDCSIT were golden due to company of entire classmates who have supported me whenever I was in need of their help and co-operation.

Love, affection, benevolence and inspiration from my family members have always enlightened my life to achieve those miracles those I couldn't even have thought of it. Without them, I would never have been what I am now. Thank you my little brother Bibek for always being behind with me and ready to help in any moments and my sisters Bhawana, Kamana and Anjana for your wonderful love, unconditional support and inspiration who have helped me to pave the path for successful journey in my academic pursuits.

Ananda K.C.
kcananda@gmail.com
December, 2012

ABSTRACT

Sentence completion is a real time ubiquitous feature directed to predict a succeeding words sequence, an appropriate completion of a given initial text fragment. Sentence completion able a user to retrieve desired information with little knowledge over exact keywords and with least typing efforts. Under statistical method, this work will deal with N-gram method to predict the remaining part of sentence for Nepali language using Viterbi as a decoding algorithm. By analyzing the result of this work, Trigram Prediction Model is more accurate than Bigram Prediction Model. To get the best result, this work recommends taking a large corpus with sufficient repetition of words.

TABLE OF CONTENTS

LIST OF FIGURES.....	v
LIST OF TABLES	vi
LIST OF ABBREVIATION	vii
1 INTRODUCTION.....	1
1.1 Introduction.....	1
1.1.1 The Challenge of Natural Language Processing	1
1.1.2 Areas of Natural Language Processing.....	2
1.1.2.1 Natural Language Understanding (NLU).....	2
1.1.2.2 Natural Language Generation (NLG)	2
1.2 Motivation.....	2
1.3 Sentence Completion.....	3
1.4 Objectives.....	4
1.5 Organization of Thesis.....	4
2 BACKGROUND AND PROBLEM DEFINITION	5
2.1 Background	5
2.1.1 Natural Language Processing	5
2.1.2 Levels of Natural Language Processing	5
2.1.2.1 Phonology	5
2.1.2.2 Morphology.....	6
2.1.2.3 Lexical	6
2.1.2.4 Syntactic.....	6
2.1.2.5 Semantic	7
2.1.2.6 Discourse.....	7
2.1.2.7 Pragmatic	7
2.1.3 Approaches to Natural Language Processing.....	7
2.1.3.1 Symbolic (Knowledge-Based)	7
2.1.3.2 Statistical (Stochastic).....	8
2.1.3.3 Connectionist Approach.....	8
2.1.4 Major Application of Natural Language Processing	8
2.1.5 Corpus linguistics.....	9
2.1.6 Statistical NLP.....	10

2.1.7 N-Gram	10
2.1.7.1 Bigram.....	11
2.1.7.2 Trigram	12
2.2 Problem Definition.....	12
3 LITERATURE REVIEW	14
3.1 Literature Review and Related Works	14
4 IMPLEMENTATION	18
4.1 Framework for Sentence Completion	18
4.1.1 Preprocessing Step	19
4.2 Bigram/Trigram Language Model using Backoff Smoothing	19
4.3 Viterbi Algorithm.....	21
4.3.1 Tracing of the Viterbi algorithm	23
4.4 SRILM	28
4.5 JAVA Programming Language	31
5 TESTING AND ANALYSIS	33
5.1 Nepali Corpus Data Statistics	33
5.2 Training and Testing Data	33
5.3 Input and Output	35
5.3.1 Snapshot of the output of the trigram and bigram model.....	38
5.4 Analysis of the Prediction Model.....	40
5.4.1 Calculation of Precision, Recall And F-Score of Bigram Prediction Model.....	40
5.4.2 Calculation of Precision, Recall and F-Score of Tri-gram Prediction Model.....	41
5.4.3 Comparison between Bigram and Trigram Prediction Model	41
5.5 Discussion of the Result	41
6 CONCLUSION AND RECOMMENDATION.....	43
6.1 Conclusion	43
6.2 Recommendation.....	43
References.....	44
Appendix	47

LIST OF FIGURES

Figure 2.1: Types of Data for Linguistic Analysis.....	9
Figure 3.1: Combination of Multiple Experts.....	16
Figure 4.1: Framework for Sentence Completion.....	19
Figure 4.2: Vocab Prepossessing.....	20
Figure 4.3: Corpus Prepossessing.....	20
Figure 4.4: Unigram Model.....	30
Figure 4.5: Bigram Model.....	30
Figure 4.6: Trigram Model.....	31
Figure 5.1: Corpus Sample.....	33
Figure 5.2: Snapshot of Bigram Output.....	38
Figure 5.3: Snapshot of Trigram Output.....	39
Figure 5.4: Comparison between Bigram and Trigram Prediction Model.....	41

LIST OF TABLES

Table 4.1: Transition Matrix of 8 words from corpus.....	23
Table 4.2: Viterbi Matrix Initialization.....	24
Table 4.3: Backtrack Matrix Initialization.....	24
Table 4.4: 1st Iteration to find Viterbi Matrix.....	24
Table 4.5: Viterbi matrix-After 1st recursion.....	25
Table 4.6: Backtrack Matrix after 1st recursion.....	25
Table 4.7: 2nd Iteration to find Viterbi Matrix.....	25
Table 4.8: Viterbi Matrix-After 2st recursion.....	26
Table 4.9: Backtrack Matrix after 2nd recursion.....	26
Table 4.10: 3rd Iteration to find Viterbi matrix.....	26
Table 4.11: Viterbi matrix-After 3rd recursion	27
Table 4.12: Backtrack Matrix after 3rd recursion	27
Table 4.13: Final Viterbi Matrix.....	27
Table 4.14: Final Backtrack Matrix.....	27
Table 5.1: Test Data.....	33-35
Table 5.2: Input and Output of the Prediction Model.....	35-37

LIST OF ABBREVIATION

NLU	Natural Language Understanding
NLP	Natural Language Processing
NLG	Natural Language Generation
IOLA	Ideal On-line Learning Algorithm
HMM	Hidden Markov Model
SLM	Statistical language modeling
LM	Language Model
MLE	Maximum Likelihood Estimation
SRILM	Stanford Research Institute Language Modeling
JVM	Java Virtual Machine

CHAPTER 1

INTRODUCTION

1.1 Introduction

Natural Language Processing (NLP) is meant for any attempt that helps the machine to understand the natural language and to generate the natural language. While researching NLP, it is to be considered that there are different types of language involved. Two different types of language can be distinguished : formal language and natural language[1].

Formal Language

Now, a formal language can be defined as a set of strings, this definition is too general to distinguish it from natural language. However, formal language does possess the property of being defined in a strict way. All sentences that are defined in such a formal language should meet the requirements as set in the definition of the language. This ensures that every sentence is set up in a standard way, which creates the possibility for computers to process these sentences.

Natural Language

Natural languages are the ones which mostly used by people to communicate. Although natural languages can differ a lot, they all share the same property of being natural. This is because these languages are not strictly bound by definitions, and so there is a natural aspect about them. For example, there are lots of pieces of language that are never used on paper, but are heavily used as spoken language in daily life violating the standards set by rules of grammar.

1.1.1 The Challenge of Natural Language Processing

Because sentences of natural language are not set up in a standard, it is very difficult for computers to understand what such a sentence states. Computers are only capable of processing language following a standard pattern. Therefore, in order to understand natural language, computers must somehow translate the natural language sentences into formal language sentences. Exactly that is the challenge which inspires scientists to research these issues within the field of natural language processing.

1.1.2 Areas of Natural Language Processing

In this section the different areas that are part of the broader term NLP are discussed. Natural language understanding is the subject of Section 1.1.2.1 whereas Natural language generation is of Section 1.1.2.2.

1.1.2.1 Natural Language Understanding (NLU)

Within the field of NLP, natural language understanding can be identified, which according to [9] is associated with the more ambitious goal of having a computer system actually comprehend natural language as a human being might or closer. Therefore the term NLU is used for the translation of human language into a format that is analyzable.

1.1.2.2 Natural Language Generation (NLG)

NLG can be defined as the subfield of artificial intelligence and computational linguistics that is concerned with the construction of computer systems that can produce understandable texts in English or other human languages from some underlying nonlinguistic representation of information[2].

Each sentence in formal language is formulated in a structured and standardized way. NLG can play an important role in the process of making a person understand these formal languages, by translating it into a language the person can understand.

1.2 Motivation

Sentence completion had been a vast research area for many other languages but due to technological and resources limitation, it is a new but with much vast scope in Nepali language. The usefulness and prospect of the NLP research in Government offices had initially intrigued me. Various minor but time consuming works in offices like letter typing, interacting with owns data warehouse, etc. can drastically increase the effectiveness and efficiency of available resources.

Another scope of Sentence completion lies within medical sciences. Various neurological disorders like “Motor Aphasia” in which a person cannot verbalize his/her thought can be benefitted from such research greatly improving life quality.

Sentence completion can be a milestone in various language processing aspects like grammar checking, machine level understanding etc. As sentence completion presents itself with many interesting and real life scopes, I am motivated to contribute in the understanding and research of this area.

1.3 Sentence Completion

Sentence prediction, a ubiquitous feature, is directed to predict which words sequence is likely to succeed given initial text fragment. It is a real time feature of returning appropriate completion. Sentence completion ables a user to retrieve desired information with little knowledge over exact keywords and with least typing efforts.

Not surprisingly, sentence prediction has found wide adoption as a feature in a variety of application. This feature is available today in program editors such as Visual Studio, command shells such as the Unix Shell, search engines such as Google, Yahoo and Desktop search. Sentence completion is also gaining popularity for mobile devices since it can assist the users in keying in contacts and text messages [3].

Consider a user wants to type “युगकवि भनेर” the system may suggest “कसलाई चिनिन्छ”. By suggesting words in any given context, word completion can assist in fast composition of well-formed text.

There are different methods for sentence prediction such as Graph completion method, Information retrieval method, Confabulation method, Statistical method, etc. Under statistical method, this work will deal with N-gram method to predict the remaining part of sentence for Nepali language.

The premise behind using N-gram language models for sentence prediction is the idea that a word is primarily dependent on previous few words. In sentence prediction, when a sequence of words is seen, the N-gram model raises the question “What words have I seen in training that followed these ones?” To do this, an N-gram model is built from some training data by recording how often each word follows a sequence of words. The number of words in the prior sequence determines the order of the N-gram model. If only the previous two words are considered, then it is called a trigram model. Similarly, for one word it is called a bigram model and a model that ignores the previous word is called a unigram model[4].

1.4 Objectives

The objectives of this work are

- To predict succeeding text fragment for sentence completion using N gram Statistical method with backoff smoothing.
- To analyze and compare the performance of backoff smoothed bigram and trigram language model with Viterbi algorithm.

1.5 Organization of Thesis

The rest of this thesis is organized as: Chapter 2 gives a brief discussion of basic concept related to this work, Chapter 3 is a survey of the major existing solution of sentence completion problem, Chapter 4 details the implementation of the Sentence completion using Smoothing Bigram/Trigram model using Katz backoff smoothing with Viterbi Algorithm, Chapter 5 presents Sentence completion results, and Chapter 6 concludes the thesis, summarizing its achievements and recommendations.

CHAPTER 2

BACKGROUND AND PROBLEM DEFINITION

2.1 Background

2.1.1 Natural Language Processing

NLP began in the 1950s as the intersection of artificial intelligence and linguistics originally distinct from text information retrieval (IR), which employs highly scalable statistics-based techniques to index and search large volumes of text efficiently. With time, however, NLP and IR have converged somewhat and also other various diverse fields.

The aim of NLP is studying problems in the automatic generation and understanding of natural language. A Natural Language is any of the languages naturally used by humans, i.e. not an artificial or machine language such as a programming language like C language, Java, Perl etc. Cognitive and linguistic motivation is to gain a better insight into how humans communicate using natural language. Technologically it is motivating to build intelligent computer systems such as machine translation systems, natural language interfaces to databases, man-machine interface to computers in general, speech understanding systems, text analysis and understanding systems, computer aided instruction systems, systems that can read and understand printed or handwritten text.

2.1.2 Levels of Natural Language Processing

The most clarifying method for presenting what actually happens within a Natural Language Processing system is by concept of the “levels of language” approach, also referred to as the synchronic model of language distinguished from the earlier sequential model, which hypothesizes that the levels of human language processing follow one another in a strictly sequential manner[5].

2.1.2.1 Phonology

Phonology is a branch of linguistics concerned with the systematic organization of sounds in languages. Classically it has focused largely on study of the systems of phonemes in particular languages, but it may also cover any linguistic analysis either at a level beneath the word or at all levels of language where sound is considered to be structured for conveying

linguistic meaning. Phonology also incorporates the study of equivalent organizational systems in sign languages[6]. This level deals with the interpretation of speech sounds within and across words. In an NLP system that accepts spoken input, the sound waves are analyzed and encoded into a digitized signal for interpretation by various rules or by comparison to the particular language model being utilized[7].

2.1.2.2 Morphology

This level deals with the componential nature of words, which are composed of morphemes – the smallest units of meaning. For example, the word preregistration can be morphologically analyzed into three separate morphemes: the prefix “pre”, the root “registra”, and the suffix “tion”. Since the meaning of each morpheme remains the same across words, an unknown word can be broken down into its constituent morphemes so that humans can understand its meaning. Similarly, an NLP system can recognize the meaning conveyed by each morpheme in order to gain the meaning. For example, adding the suffix –ed to a verb, conveys that the action of the verb occurred in past[5].

2.1.2.3 Lexical

At this level, NLP systems, interpret the meaning of individual words. Several types of processing contribute to word-level understanding – the first of these being assignment of a single part-of-speech tag to each word. In this processing, words that can function as more than one part-of-speech are assigned the most probable part-of-speech tag based on the context in which they occur. Additionally at the lexical level, those words that have only one possible sense or meaning can be replaced by a semantic representation of that meaning. The nature of the representation varies according to the semantic theory utilized in the NLP system[5].

2.1.2.4 Syntactic

This level specializes on analyzing the words in a sentence so as to uncover the grammatical structure of the sentence. This requires both a grammar and a parser. The output of this level of processing is a representation of the sentence that shows the structural dependency relationships between the words. There are several grammars that can be utilized, and which will, in turn, impact the choice of a parser.

2.1.2.5 Semantic

This is the level at which most people think meaning is determined, however, as it can be seen in the above defining of the levels, it is all the levels that contribute to meaning. Semantic processing determines the possible meanings of a sentence by focusing on the interactions among word-level meanings in the sentence.

2.1.2.6 Discourse

As syntax and semantics work with sentence-length units, the discourse level of NLP works with units of text longer than a sentence meaning it does not interpret multi-sentence texts as just concatenated sentences, but each can be interpreted singly. Rather, discourse specializes on the properties of the text as a whole that convey meaning by making connections among component sentences.

2.1.2.7 Pragmatic

The level is to explain how extra meaning is read into texts without being actually encoded in them. The purposeful use of language in situations is concerned. Above the contents of the text are used for understanding requiring a lot of world knowledge. Knowledge bases and inferencing modules are utilized in some NLP applications.

2.1.3 Approaches to Natural Language Processing

Natural language processing approaches fall roughly into four groups: symbolic, statistical, connectionist, and hybrid. Symbolic and statistical approaches have coexisted since the beginning. Connectionist NLP work first appeared in the 1960's. For a long time, symbolic approaches dominated the field. In the 1980's, statistical approaches regained popularity as a result of the availability of critical computational resources and the need to deal with broad, practical contexts. Connectionist approaches also recovered from earlier criticism by demonstrating the utility of neural networks in NLP.

2.1.3.1 Symbolic (Knowledge-Based)

Based on explicit representation of facts about language through well-understood knowledge representation schemes and associated algorithms [8] symbolic approaches is seen in logic or rule-based systems. In logic-based systems, the symbolic structure is commonly in the form of logic propositions. Manipulations of such structures are defined by inference procedures

that are generally truth preserving. Rule-based systems usually consist of a set of rules, an inference engine, and a workspace or working memory. Knowledge is represented as facts or rules in the rule-base. The inference engine repeatedly identifies a rule whose condition is satisfied and executes the rule.

2.1.3.2 Statistical (Stochastic)

Stochastic models ability is to work well even in the presence of incomplete linguistic knowledge about an application domain. Such models thrive on the inherent inflexibility and fragility of symbolic models, which stem from our lack of complete understanding of many linguistic phenomena – an essential prerequisite to developing successful symbolic models[8].A frequently used statistical model is the Hidden Markov Model (HMM) inherited from the speech community. Statistical approaches have typically been used in works such as speech recognition, parsing, part-of-speech tagging, collocations, statistical machine translation, statistical grammar learning, and lexical acquisition.

2.1.3.3 Connectionist Approach

Similar to the statistical approaches, connectionist approaches also develop generalized models from examples of linguistic phenomena. What separates connectionism from other statistical methods is that it combines statistical learning with various theories of representation – thus the connectionist representations allow transformation, inference, and manipulation of logic formulae. In addition, in connectionist systems, linguistic models are difficult to observe due to the fact that connectionist architectures are less constrained than statistical ones[9].

2.1.4 Major Application of Natural Language Processing

The following is a list of some of the most commonly researched tasks in NLP. Some of these tasks have direct real-world applications, while others more commonly serve as subtasks that are used to aid in solving larger tasks. What distinguishes these tasks from other potential and actual NLP tasks is not only the volume of research devoted to them but the fact that for each one there is typically a well-defined problem setting, a standard metric for evaluating the task, standard corpora on which the task can be evaluated, and competitions devoted to the specific task.

- Part-of-speech tagging
- Information retrieval

- Machine translation
- Morphological segmentation
- Question answering
- Relationship extraction
- Sentiment analysis
- Speech segmentation
- Word segmentation
- Word sense disambiguation
- Natural language generation
- Machine translation
- Named entity recognition
- Automatic summarization
- Discourse analysis

2.1.5 Corpus linguistics

Corpus linguistics is a method of carrying out linguistic analyses. It can be used for the investigation of many kinds of linguistic questions. Also as it has been shown to have the potential to yield highly interesting, fundamental, and often surprising new insights about language, it has become one of the most wide-spread methods of linguistic investigation in recent years. Corpus linguistics thus is the analysis of naturally occurring language on the basis of computerized corpora. Usually, the analysis is done with the help of the computer, i.e. with specialized software, and takes into account the frequency of the phenomena investigated. Roughly, four types of data for linguistic analysis can be distinguished.

Data gained by intuition	The researcher's own intuition ("introspection")
	Other people's ("informant's") intuition
Naturally occurring language	Randomly collected texts or occurrences ("anecdotal evidence")
	Systematic collections of texts ("corpora")

Figure 2.1: Types of data for linguistic analysis

A corpus can be defined as a systematic collection of naturally occurring texts. "Systematic" means that the structure and contents of the corpus follows certain extra linguistic principles.

For example, a corpus is often limited to certain text types, to one or several varieties of Nepali, and to a certain time span.

2.1.6 Statistical NLP

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 incorporates speech recognition, machine translation, document classification and routing, optical character recognition, information retrieval, handwriting recognition, spelling correction, and many more[10].The use of statistical methods to natural language processing has been strikingly successful over the past two decades. The wide availability of text and speech corpora has played a major role in their success since, as for all learning techniques, these methods heavily rely on data. Many of the components of complex natural language processing systems, e.g., text normalizers, morphological or phonological analyzers, part-of-speech taggers, grammars or language models, acoustic Hidden-Markov Models (HMMs), pronunciation models, and context dependency models are statistical models derived from large datasets using modern learning techniques. These models are often given as weighted automata or weighted finite-state transducers either directly or as a result of the approximation of more complex models.

In the current literature on natural language processing (NLP), a distinction is often made between “rule-based” and “statistical” methods for NLP. However, it is seldom made clear what the terms “rule-based” and “statistical” really refer to in this connection otherwise it is said to be stochastic. Furthermore, a stochastic model is said to be probabilistic or statistical, if its representation is from the theories of probability or statistics, respectively.

2.1.7 N-Gram

Probabilities are based on counting things. Counting of things in natural language is based on a corpus (plural corpora), an on-line collection of text or speech. The goal is to compute the probability of the given word “w” with given some history “h”. If the history is “जेठ चौधमा संविधान जारी” then the probability the next word is “हुन” is

$$P(\text{हुन}|\text{जेठ चौधमा संविधान जारी}) \quad (2.1)$$

One way to estimate the probability is from relative frequency counts. From the very large corpus count the no of times “जेठ चौधमा संविधान जारी” and “जेठ चौधमा संविधान जारी हुन” occurred. The probability is calculated as

$$P(\text{हुन}|\text{जेठ चौधमा संविधान जारी}) = \frac{C(\text{जेठ चौधमा संविधान जारी हुन})}{C(\text{जेठ चौधमा संविधान जारी})} \quad (2.2)$$

While this method of estimating probabilities directly from counts works fine in many cases, it turns out that even the corpus isn't big enough to give us good estimates in most cases. This is because language is creative; new sentences are created all the time, and this won't always be able to count entire sentences. A sequence of N words is represented as $W_1, W_2, W_3, \dots, W_n$ or W_1^n . For the joint probability of each word in a sequence having a particular value, this work will use $P(W_1, W_2, \dots, W_n)$. Now the probability of the entire sequence is calculated by using the chain rule of probability as

$$\begin{aligned} P(W_1, \dots, W_n) &= P(W_1)P(W_2|W_1)P(W_3|W_1^2) \dots P(W_n|W_1^{n-1}) \\ &= \prod_{k=1}^n P(W_k|W_1^{k-1}) \end{aligned} \quad (2.3)$$

The intuition of the N-gram model is that instead of computing the probability of a word given its entire history, the history is approximated by just the last few words. Thus the general equation for this N-gram approximation to the conditional probability of the next word in a sequence is

$$P(W_n|W_1^{n-1}) = P(W_n|W_{n-N+1}^{n-1}) \quad (2.4)$$

2.1.7.1 Bigram

The bigram model approximates the probability of a word given all the previous words $P(W_n|W_1^{n-1})$ by the conditional probability of the preceding word $P(W_n|W_{n-1})$. Given the bigram assumption for the probability of an individual word, the probability of a complete word sequence can be calculated using equation 2.3 as

$$P(W_n|W_1^{n-1}) = \prod_{k=1}^n P(W_k|W_{k-1}) \quad (2.5)$$

The simplest and most intuitive way to estimate probabilities is called Maximum Likelihood Estimation (MLE). For the general case of MLE, N-gram parameter estimation

$$P(W_n|W_{n-N+1}^{n-1}) = \frac{C(W_{n-N+1}^{n-1} W_n)}{C(W_{n-N+1}^{n-1})} \quad (2.6)$$

So for the bigram model the probability is derived from equation 2.6 as

$$P(W_n|W_{n-1}) = \frac{C(W_{n-1} W_n)}{C(W_{n-1})} \quad (2.7)$$

2.1.7.2 Trigram

The trigram model approximates the probability of a word given all the previous words $P(W_n|W_1^{n-1})$ by the conditional probability of the preceding two words $P(W_n|W_{n-2}W_{n-1})$. Given the bigram assumption for the probability of an individual word, the probability of a complete word sequence can be calculated using equation 2.3 as

$$P(W_n|W_1^{n-1}) = \prod_{k=1}^n P(W_k|W_{k-2}W_{k-1}) \quad (2.8)$$

So, for the trigram model the probability is calculated from equation 2.6 as

$$P(W_n|W_{n-2}^{n-1}) = \frac{C(W_{n-2}W_{n-1}W_n)}{C(W_{n-2}W_{n-1})} \quad (2.9)$$

2.2 Problem Definition

If we have given the text fragment W_1, \dots, W_t , then the problem is to find the most likely word sequence W_{t+1}, \dots, W_{t+T} to complete the sentence for the given text fragment.

Sentence prediction facilitates text entry by suggesting sequence of words. If the desired text fragment is suggested, the user can select with a mouse click or a keystroke, thereby saving the effort of typing the remaining words. Otherwise, the user can continue typing while the software continues to display new sequence of words based on the input.

Sentence completion in search is one of the most popular features to be found in wide variety of application environments. As the user searches for the text, suggestions based on typed content are returned[11].For example when the user types “जेठ चौधमा” then text fragment “संविधान जारी हुन सकेन” is given by the system for sentence completion.

It has been widely adopted in text & code editors, and in almost every possible browsing Graphical User Interfaces (GUIs), -where user tries to input information. In absence of Sentence completion, a user would either type entire string or part of it which may or may not return the desired results[12].

It is also used by human translator who will choose a given foreign sentence[13, 14].In translation from English to Nepali such as “I love my mother.” After a user writes “म मेरो आमालाई” then “माया गर्छु” option should be provided. Then the user will select the right one.

The sentence prediction also assist user in search engine[3, 12].When the user types “नेपालमा कम्प्युटरको” then the sentence completion system provides “कति पर्छ”.Also when the user is typing, it can save time to type the remaining words. For example: when the input is “मुनामदनका लेखक” then the purposed system gives the word “लक्ष्मी प्रसाद देवकोटा हुन्” then it can save time and also typing effort. Also for the user with little knowledge of Nepali language, this feature can assist by providing possible succeeding words for completion.

CHAPTER 3

LITERATURE REVIEW

3.1 Literature Review and Related Works

Through analysis of the predictability of sequence of letters, it was revealed that written English has a high degree of redundancy [15]. The ordinary literary English has long range statistical effects which reduce the entropy to something of the order of one bit per letter, with a corresponding redundancy of roughly 75% [15]. In addition, their redundancy may be still higher when structure extending over paragraphs, chapters, etc. is considered. The parameters in question become more erratic and uncertain with the increased in lengths and in turn they depend more critically on the type of text involved. Therefore, based on this finding, it is obvious to ask whether the system can support users in the process of writing text which predict the intended next words, keystroke or sentence [13].

It has been studied that “Interactive keyboard” uses the sequence of past keystrokes to predict the most likely succeeding keystroke [16]. This approach [16] of keystrokes was determined statistically. After that, almost all of the predictive models developed.

Normally, the Statistical methods imply words based on:

- a) Frequency, either in the context of relevant corpora or what the user has typed in the past;
or
- b) Recently, where suggested words are those the user has most recently typed.

This method has its own shortcomings but it reduces keystrokes and increase efficiency. Though it was designed for general purpose, this computer interface has great scope to enhance the ease and rate of communication especially for physically limited people.

Identifying user-dependent information, which can be automatically collected, helps to build a user model by which

- a) To predict what the user wants to do next and
- b) To do relevant processing

This information is often relational which can be best represented by a set of directed graphs. A machine learning technique called graph-based induction (GBI) efficiently extracts regularities from such data. Based on which a user-adaptive interface is built which can predict next command, generate scripts and pre fetch files in a multi task environment. The core of GBI is pair wise chunking. This approach shows how this simple mechanism can be applied to the top down induction of decision trees for nested attribute representation as well as finding frequently occurring patterns in a graph. The results, activated by the user commands, show that the dependency analysis of computational processes is really useful to build a behavior model and increase prediction accuracy[17].

The approach[18] where the recent actions strongly affect future actions than older actions propose the following description of an Ideal On-line Learning Algorithm (IOLA). In order to contain the desirable characteristics of the best algorithms, an IOLA would:

- a) Have predictive accuracy at least as good as the best known resource-unlimited methods,
- b) Operate incrementally,
- c) Be affected by all events,
- d) Not need to retain a copy of the user's full history of actions,
- e) Output a list of predictions, sorted by confidence,
- f) Adapt to changes to the target concept,
- g) Be fast enough for interactive use,
- h) learn by passively watching and
- i) Apply even in the absence of domain knowledge.

This approach[19] has discovered a variety of techniques to improve its accuracy, including a "mixture of experts" model as shown in figure 3.1.

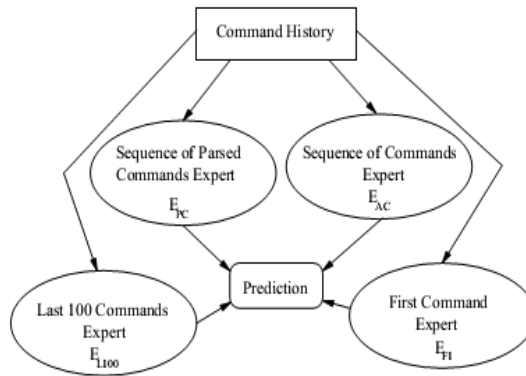


Figure 3.1: Combination of Multiple Experts[19]

On a series of experiments with real world data, this approach show

- a) Within 50% of the time, the simple system can correctly predict the user's next command and can perform robustly across a range of different users and
- b) It is extremely difficult to further improve this result

A problem of predicting the next action of a user is frequently dealt by the area of human computer interface design. Unix shell is developed which predicts the command stubs that a user is most likely to enter, given the current history of entered commands[17, 18]. This idea has been elaborated which predicts whole command lines [19]. To predict the next keystrokes, the Reactive Keyboard [16] uses length-k modeling which is based on the previous keystrokes typed. Though it is similar to work[19], it is also predicting the next complete command line which is based on the previous command lines typed, rather than a keystroke at a time.

A number of different models have been purposed for the prediction of words, keystroke or sentence for example: An indexing which efficiently retrieves the sentence from a collection was developed that is most similar to a given initial fragment[20]. To address the sentence completion problem, the cogent confabulation model was used[21]. Cogent confabulation, a bio-inspired model, extracts posterior probabilities among objects at the symbol level. Also, a graph mining methodology can be used to address the same sentence completion problem [12].

The email programs suggestions which come from auto completion database, is created from users mailing history and contacts list. This characteristic support users to fill up address field just after they typed few characters in "TO", "CC" or "BCC" field of an

outgoing email. Beyond this, there are many other applications that uses different forms auto complete as per the relevance. Just like the text processors generate their suggestions list based on previous search history or from a predefined list. However, in majority of text processors this suggestions are only limited to word length. Even though they might not actually exist in the text which user is trying to get useful information from, are based on the typed characters[12].

Several scholars has conducted many research on statistical language model and its associated corpus [4]. Likewise the grammatical judgment task is also used for Sentence completion task[22].

Statistical language model [23, 24] mainly N-gram language model [11, 18, 20, 22, 25, 26] having backoff smoothing[4]has been used in sentence completion. In addition, this models also uses application of the Viterbi principle[27] for sentence completion.

CHAPTER 4

IMPLEMENTATION

4.1 Framework for Sentence Completion

Sentence completion flowchart is given in Figure 4.1. This describes top level data flow diagram of sentence completion problem, used in this work. The proposed system framework consists of three steps; preprocessing, train the data using bigram/trigram language model with back off smoothing and find the most probable sequence using Viterbi algorithm.

Preprocessing sub engine has Nepali corpus as input. After preprocessing, only important data is stored. Training the corpus is the most powerful and is the heart of Sentence completion Model. After training corpus, the most probable sequence for the input text fragment is find out using Viterbi algorithm.

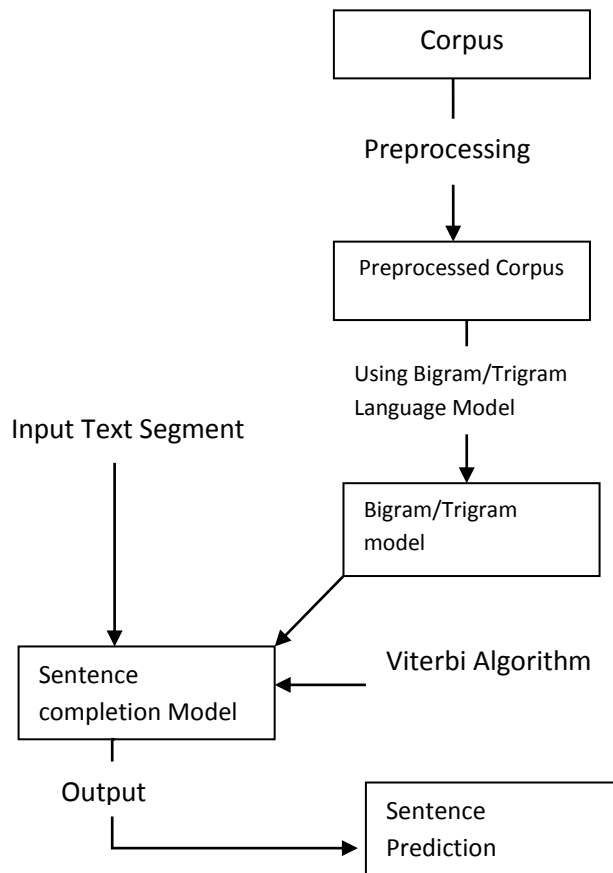


Figure 4.1:- Framework for Sentence Completion

4.1.1 Preprocessing Step

Preprocessing is one of the important steps in sentence completion. It helps to make the raw data suitable for analysis by improving the data quality. Preprocessing step makes input ready for the Stanford Research Institute Language Modeling (SRILM). In preprocessing step two files are created. The input format for the SRILM toolkit is shown in figure 4.2 and 4.3. Accordingly the java program is written to make the necessary file.

बनेको
बोकी
भएपनि
अन्त्यमा
बास्केटबलमा
आफू
खेलकुदको
आस
राति
अखिल
तिने
स्क्रिनमा
हुँदैछन्
रेफ्रीले
कतबका
इटालीका
प्रतिष्ठित
एकपछि
गराउँदै
महत्वपूर्ण
टिममाथि

Figure 4.2: Vocab Prepossessing

आमी अफिसस क्लबमा उनले तेस्रो दिन १ अन्डर ७१ स्कोर गरेसजिनले तेस्रो चरणमा युनिभर्सलका निशान हुंगेललाई पराजित गरे भने चौथोमा पुल्लोक इन्जिनियरिङका आशिष आचार्यसँग बराबरी खेले तीन वर्षयता पहिलोपल्ट उपाधि जित्ने अभियानमा रहेका उनी समग्रमा ३ अन्डर २ सय १३ मा छन् दुई वर्षअघि नेपाल नम्बर एक खेलाडी विकास श्रेष्ठलाई हराएर सनसनी मच्चाएका दिपेश धामी अहिले नेपाली ब्याडमिन्टनको सुखद भविष्य कोने सफल हुँदैछन् अन्तिम होलमा दुई स्ट्रोक जोगाउँदै इगल आत्मसात् गरेपछि रवि खड्का घनश्याम ओपन गल्फमा लगातार तेस्रो दिन शीर्षस्थानमा रहन सफल भएका छन् ५६ खेलाडी सहभागी प्रतियोगीको ५४औं स्थानमा आउँदा उनी राष्ट्रिय कीर्तिमान सूधार गर्ने असफल भइन् नेपाली राष्ट्रिय टिमले भारतीय फुटबलको प्रतिष्ठित प्रतियोगिता नेहरू कप खेल्ने भएका छ अन्तिम होलमा दुई स्ट्रोक जोगाउँदै इगल आत्मसात् गरेपछि रवि खड्का घनश्याम ओपन गल्फमा लगातार तेस्रो दिन शीर्षस्थानमा रहन सफल भएका छन् औपचारिक रुपमा उदघाटन सुरु नहुँदै ओलम्पिक विवादमा परेको छ उदघाटन नहुँदै सुरु भएका केही खेलहरूमध्ये महिला फुटबल खेलमा गएराति आयोजकहरूले उत्तरकोरियाको झण्डा स्क्रिनमा देखाउनु पने ठाउँमा दक्षिण कोरियाको देखाउन प्रयोग थिए

Figure 4.3: Corpus Prepossessing

4.2 Bigram/Trigram Language Model using Backoff Smoothing

The major problem with the maximum likelihood estimation process for training the parameters of an N-gram model is sparse data. The problem sparse data is caused by the fact that maximum likelihood estimate was based on a particular set of training data. Any N-gram that occurred a sufficient number of times have a good estimate of its probability. But

because any corpus is limited, some perfectly acceptable nepali word sequences are bound to be missing from it. This missing data means that the N-gram matrix for any given training corpus is bound to have a very large number of cases of putative “zero probability N-grams” that should really have some non-zero probability. Furthermore, the MLE method also produces poor estimates when the counts are non-zero but still small.

The way to solve this “zero probability N-grams” problem is Smoothing. Not only do smoothing methods generally prevent zero probabilities, but they also attempt to improve the accuracy of the model as a whole. Whenever a probability is estimated from few counts, smoothing has the potential to significantly improve estimation.

One way to solve this problem is called Katz backoff. In a Katz backoff N-gram model, if the N-gram concerned has zero counts, it is approximated by backing off to the (N-1)-gram. In this backing off a history that has some counts is reached.

$$P_{katz}(W_i|W_{i-2}W_{i-1}) = \begin{cases} P^*(W_i|W_{i-2}W_{i-1}), & \text{if } C(W_{i-2}W_{i-1}W_i) > 0 \\ \alpha(W_{i-2}W_{i-1})P(W_i|W_{i-1}), & \text{elseif } C(W_{i-1}W_i) > 0 \\ \alpha(W_i)P^*(W_i), & \text{otherwise} \end{cases} \quad (4.1)$$

$$P_{katz}(W_i|W_{i-1}) = \begin{cases} P^*(W_i|W_{i-1}), & \text{if } C(W_{i-1}W_i) > 0 \\ \alpha(W_{i-1})P^*(W_i), & \text{otherwise} \end{cases} \quad (4.2)$$

Here P^* is defined as the discounted estimate of the conditional probability of an N-gram.

$$P^*(W_n|W_{n-N+1}^{n-1}) = \frac{C^*(W_{n-N+1}^n)}{C(W_{n-N+1}^{n-1})} \quad (4.3)$$

$$C^* = \frac{(C+1)\frac{N_{c+1}}{N_c} - C\frac{(k+1)N_{k+1}}{N_1}}{1 - \frac{(k+1)N_{k+1}}{N_1}}, \text{ for } 1 \leq C \leq k \quad (4.4)$$

By discounting, this will leave some probability mass for the lower order N-gram, which is then distributed by the α weight and the total amount is represent of left-over-probability mass by the function β , a function of the (N-1)-gram context.

$$\beta(W_{n-N+1}^{n-1}) = 1 - \sum_{W_n: C(W_{n-N+1}^n) > 0} P^*(W_n|W_{n-N+1}^{n-1}) \quad (4.5)$$

$$\begin{aligned}
\alpha(W_{n-N+1}^{n-1}) &= \frac{\beta(W_{n-N+1}^{n-1})}{\sum_{W_n: C(W_{n-N+1}^n)=0} P_{katz}(W_n|W_{n-N+2}^{n-1})} \\
&= \frac{1 - \sum_{W_n: C(W_{n-N+1}^n)>0} P^*(W_n|W_{n-N+1}^{n-1})}{1 - \sum_{W_n: C(W_{n-N+1}^n)>0} P^*(W_n|W_{n-N+2}^{n-1})}
\end{aligned} \tag{4.6}$$

4.3 Viterbi Algorithm

The problem is to find the most likely word sequence $W_{t+1} \dots W_{t+T}$ given an initial word sequence $W_1 \dots W_t$.

$$\operatorname{argmax}_{W_{t+1} \dots W_{t+T}} P(W_{t+1} \dots W_{t+T} | W_1 \dots W_t) \tag{4.7}$$

By using joint probability of the missing word the equation 4.8 can be obtained.

$$\operatorname{argmax}_{W_{t+1} \dots W_{t+T}} P(W_{t+1} \dots W_{t+T} | W_1 \dots W_t) = \operatorname{argmax}_{W_{t+1} \dots W_{t+T}} \prod_{j=1}^T P(W_{t+j} | W_1 \dots W_{t+j-1}) \tag{4.8}$$

Using N-gram model simplifies the equation 4.8 in equation 4.9

$$\operatorname{argmax}_{W_{t+1} \dots W_{t+T}} P(W_{t+1} \dots W_{t+T} | W_1 \dots W_t) = \operatorname{argmax}_{W_{t+1} \dots W_{t+T}} \prod_{j=1}^T P(W_{t+j} | W_{t+j-N+1} \dots W_{t+j-1}) \tag{4.9}$$

By identifying the recursive structure in equation 4.7, lead to Viterbi algorithm that retrieves the most likely word sequence. At first an auxiliary variable $\delta_{t,s}(w'_1, \dots, w'_N | W_{t-N+2}, \dots, W_t)$ is defined in equation 4.10; which quantifies the greatest possible probability over all arbitrary word sequences W_{t+1}, \dots, W_{t+s} followed by the word sequence $W_{t+s+1} = w'_1, \dots, W_{t+s+N} = w'_N$. In equation 4.11, the last transition is factorized and then the N-th order Markov assumption is applied. In equation 4.12, after splitting the maximization, a new random variable W'_0 for W_{t+s} is introduced. Now the recursion can be observed in equation 4.13 referring the definition of δ .

$$\begin{aligned}
&\delta_{t,s}(w'_1, \dots, w'_N | W_{t-N+2}, \dots, W_t) \\
&= \max_{W_{t+1} \dots W_{t+s}} P(W_{t+1}, \dots, W_{t+s}, w'_1, \dots, w'_N | W_{t-N+2}, \dots, W_t)
\end{aligned} \tag{4.10}$$

$$\begin{aligned} & \delta_{t,s}(W'_1, \dots, W'_N | W_{t-N+2}, \dots, W_t) \\ &= \max_{W'_{t+1}, \dots, W'_{t+s}} P(W'_N | W'_1, \dots, W'_{N-1}) P(W_{t+1}, \dots, W_{t+s}, W'_1, \dots, W'_{N-1} | W_{t-N+2}, \dots, W_t) \end{aligned} \quad (4.11)$$

$$\begin{aligned} & \delta_{t,s}(W'_1, \dots, W'_N | W_{t-N+2}, \dots, W_t) \\ &= \max_{W'_0} \max_{W'_{t+1}, \dots, W'_{t+s-1}} P(W'_N | W'_1, \dots, W'_{N-1}) P(W_{t+1}, \dots, W_{t+s} = W'_0, W'_1, \dots, W'_{N-1} | W_{t-N+2}, \dots, W_t) \end{aligned} \quad (4.12)$$

$$\begin{aligned} & \delta_{t,s}(W'_1, \dots, W'_N | W_{t-N+2}, \dots, W_t) \\ &= \max_{W'_0} P(W'_N | W'_1, \dots, W'_{N-1}) \delta_{t,s-1}(W'_0, \dots, W'_{N-1} | W_{t-N+2}, \dots, W_t) \end{aligned} \quad (4.13)$$

Exploiting the N-th order Markov assumption, target probability in terms of δ is expressed in equation 4.14.

$$\begin{aligned} & \max_{W'_{t+1}, \dots, W'_{t+T}} P(W_{t+1}, \dots, W_{t+T} | W_{t-N+2}, \dots, W_t) \\ &= \max_{W'_1, \dots, W'_N} \delta_{t,T-N}(W'_1, \dots, W'_N | W_{t-N+2}, \dots, W_t) \end{aligned} \quad (4.14)$$

The last N words in the most likely sequence are simply the $\operatorname{argmax}_{W'_1, \dots, W'_N} \delta_{t,T-N}(W'_1, \dots, W'_N | W_{t-N+2}, \dots, W_t)$. In order to collect the preceding most likely words, an auxiliary variable φ is defined in equation 4.15 that can be determined in equation 4.16.

$$\begin{aligned} & \varphi_{t,s}(W'_1, \dots, W'_N | W_{t-N+2}, \dots, W_t) \\ &= \operatorname{argmax}_{W'_{t+s}} \max_{W'_{t+1}, \dots, W'_{t+s-1}} P(W_{t+1}, \dots, W_{t+s}, \dots, W'_N | W_{t-N+2}, \dots, W_t) \end{aligned} \quad (4.15)$$

$$\begin{aligned} & \varphi_{t,s}(W'_1, \dots, W'_N | W_{t-N+2}, \dots, W_t) \\ &= \operatorname{argmax}_{W'_0} \delta_{t,s-1}(W'_0, \dots, W'_{N-1} | W_{t-N+2}, \dots, W_t) P(W'_N | W'_1, \dots, W'_{N-1}) \end{aligned} \quad (4.16)$$

The Viterbi algorithm starts with the most recently entered word W_t and moves iteratively into future. Viterbi search is stopped when T reaches 4[13].

Viterbi java function:

```
// Viterbi Matrix Initialization
for(int j=0;j<h.size();j++)
{
    Vmatrix[j][0]=Tmatrix[1][j];
}
```

```

}
// Backtrack Matrix Initialization

for(int j=0;j<h.size();j++)
{
    Bmatrix[j][0]=1;
}
// Viterbi matrix and backtrack matrix calculation
int index=0;
for(int t=1;t<4;t++)

{
    for(int ip=0;ip<h.size();ip++)
    {
        double value=0.0;

        for(int j=0;j<h.size();j++)
        {
            if(Vmatrix[j][t-1]*Tmatrix[j][ip]>value)
            {
                value=Vmatrix[j][t-1]*Tmatrix[j][ip];
                index=j;
            }
        }

        Vmatrix[ip][t]=value;
        Bmatrix[ip][t]=index;
    }
}

```

4.3.1 Tracing of the Viterbi algorithm

Tables 4.1 represent a transition matrix of 8 words from the corpus.

	समूह	ए	को	अन्तिम	खेलमा	नेपाल	भिड्दै	छन्
समूह	0.001789938	0.08395	0.004858407	0.001279	0.003452	0.001662	3.84E-04	0.006776
ए	0.001886556	6.74E-04	0.058731487	0.001348	0.003638	0.001752	4.04E-04	0.007142
को	0.001841183	6.58E-04	0.004997502	0.010326	0.003551	0.00171	3.95E-04	0.00697
अन्तिम	0.001873347	6.69E-04	0.005084805	0.001338	0.03924	0.00174	4.01E-04	0.007092
खेलमा	0.00188778	6.74E-04	0.00512398	0.001348	0.003641	0.010876	4.05E-04	0.007147
नेपाल	0.00182362	6.51E-04	0.004949831	0.001303	0.003517	0.001693	3.91E-04	0.006904
भिड्दै	0.001529034	5.46E-04	0.004150239	0.001092	0.002949	0.00142	3.28E-04	0.391772
छन्	4.95E-05	1.77E-05	1.34E-04	3.54E-05	9.55E-05	4.60E-05	1.06E-05	1.87E-04

Table 4.1: Transition Matrix of 8 words from corpus

- I. Let the system have the input “<s> नेपाल”
- II. At first, the initialization of the Viterbi matrix is as shown in table 4.2.

0.001823620	0.000000000	0.000000000	0.000000000
0.000651293	0.000000000	0.000000000	0.000000000
0.004949831	0.000000000	0.000000000	0.000000000
0.001302586	0.000000000	0.000000000	0.000000000
0.003516984	0.000000000	0.000000000	0.000000000
0.001693364	0.000000000	0.000000000	0.000000000
0.000390776	0.000000000	0.000000000	0.000000000
0.006903708	0.000000000	0.000000000	0.000000000

Table 4.2:Viterbi Matrix Initialization

III. Backtrack matrix initialization is as shown in Table 4.3.

5	0	0	0
5	0	0	0
5	0	0	0
5	0	0	0
5	0	0	0
5	0	0	0
5	0	0	0
5	0	0	0

Table 4.3:Backtrack Matrix Initialization

IV. Recursion to calculate the Viterbi matrix 2nd column and Backtrack matrix 2nd column, this work multiplies Viterbi[0][0]* Transition[0][1], Viterbi[0][1]* Transition[1][1], Viterbi[2][0]* Transition[2][1], Viterbi[3][0]*Transition[3][1], Viterbi[4][0]* Transition[4][1], Viterbi[5][0]* Transition[5][1], Viterbi[6][0]* Transition[6][1],and Viterbi[7][0]*Transition[7][1].To fill in the Viterbi[0][1],find the maximum result from above and to fill in the backtrack[0][1],find from which index the maximum Viterbi value has been derived. The table 4.4 shows the multiplication to find the Viterbi value for 2nd column. Underlined figures are the highest figure in their respective column.

समूह	'ए'	को	अन्तिम	खेलमा	नेपाल	भिड्दै	छन्
0.000003264	<u>0.000153095</u>	0.000008860	0.000002332	0.000006295	0.000003031	0.000000699	0.000012357
0.000001229	0.000000439	<u>0.000038251</u>	0.000000878	0.000002370	0.000001141	0.000000263	0.000004652
<u>0.000009114</u>	0.000003255	0.000024737	<u>0.000051114</u>	0.000017576	0.000008463	<u>0.000001953</u>	0.000034501
0.000002440	0.000000871	0.000006623	0.000001743	<u>0.000051114</u>	0.000002266	0.000000523	0.000009238
0.000006639	0.000002371	0.000018021	0.000004742	0.000012804	<u>0.000038251</u>	0.000001423	0.000025134
0.000003088	0.000001103	0.000008382	0.000002206	0.000005956	0.000002867	0.000000662	0.000011690
0.000000598	0.000000213	0.000001622	0.000000427	0.000001152	0.000000555	0.000000128	<u>0.000153095</u>
0.000000342	0.000000122	0.000000927	0.000000244	0.000000659	0.000000317	0.000000073	0.000001294

Table 4.4:1st Iteration to Find Viterbi Matrix

V. So the Viterbi matrix after the 1st recursion is shown in table 4.5.

0.001823620	0.000009114	0.000000000	0.000000000
0.000651293	0.000153095	0.000000000	0.000000000
0.004949831	0.000038251	0.000000000	0.000000000
0.001302586	0.000051114	0.000000000	0.000000000
0.003516984	0.000051114	0.000000000	0.000000000
0.001693364	0.000038251	0.000000000	0.000000000
0.000390776	0.000001953	0.000000000	0.000000000
0.006903708	0.000153095	0.000000000	0.000000000

Table 4.5:Viterbi Matrix-After 1st recursion

VI. And the Backtrack matrix after the 1st recursion is shown in table 4.6.

5	2	0	0
5	0	0	0
5	1	0	0
5	2	0	0
5	3	0	0
5	4	0	0
5	2	0	0
5	6	0	0

Table 4.6:Backtrack Matrix after 1st Recursion

VII. At 2nd recursion, to find the Viterbi value and backtrack value for the third column the multiplication is shown in table 4.7. Underlined figures are the highest figure in their respective column.

समूह	'ए'	को	अन्तिम	खेलमा	नेपाल	भिइदै	छन्
0.000000016	<u>0.000000765</u>	0.000000044	0.000000012	0.000000031	0.000000015	0.000000003	0.000000062
<u>0.000000289</u>	0.000000103	<u>0.000008992</u>	0.000000206	0.000000557	0.000000268	<u>0.000000062</u>	<u>0.000001093</u>
0.000000070	0.000000025	0.000000191	<u>0.000000395</u>	0.000000136	0.000000065	0.000000015	0.000000267
0.000000096	0.000000034	0.000000260	0.000000068	<u>0.000002006</u>	0.000000089	0.000000021	0.000000362
0.000000096	0.000000034	0.000000262	0.000000069	0.000000186	<u>0.000000556</u>	0.000000021	0.000000365
0.000000070	0.000000025	0.000000189	0.000000050	0.000000135	0.000000065	0.000000015	0.000000264
0.000000003	0.000000001	0.000000008	0.000000002	0.000000006	0.000000003	0.000000001	0.000000765
0.000000008	0.000000003	0.000000021	0.000000005	0.000000015	0.000000007	0.000000002	0.000000029

Table 4.7:2nd Iteration to Find Viterbi Matrix

VIII. So the Viterbi matrix after the 2nd recursion is shown in table 4.8.

0.001823620	0.000009114	0.000000289	0.000000000
0.000651293	0.000153095	0.000000765	0.000000000
0.004949831	0.000038251	0.000008992	0.000000000
0.001302586	0.000051114	0.000000395	0.000000000
0.003516984	0.000051114	0.000002006	0.000000000
0.001693364	0.000038251	0.000000556	0.000000000
0.000390776	0.000001953	0.000000062	0.000000000
0.006903708	0.000153095	0.000001093	0.000000000

Table 4.8:Viterbi Matrix-After 2st Recursion

IX. And the Backtrack matrix after the 2nd recursion is shown in table 4.9.

5	2	1	0
5	0	0	0
5	1	1	0
5	2	2	0
5	3	3	0
5	4	4	0
5	2	1	0
5	6	1	0

Table 4.9:Backtrack Matrix after 2nd recursion

X. At 3rd recursion, to find the Viterbi value and backtrack value for the third column the multiplication is shown in table 4.10. Underlined figures are the highest figure in their respective column.

समूह	'ए'	को	अन्तिम	खेलमा	नेपाल	भिड्दै	छन्
0.000000001	<u>0.000000024</u>	0.000000001	0.000000000	0.000000001	0.000000000	0.000000000	0.000000002
0.000000001	0.000000001	<u>0.000000045</u>	0.000000001	0.000000003	0.000000001	0.000000000	0.000000005
<u>0.000000017</u>	0.000000006	0.000000045	<u>0.000000093</u>	<u>0.000000032</u>	0.000000015	<u>0.000000004</u>	<u>0.000000063</u>
0.000000001	0.000000000	0.000000002	0.000000001	0.000000015	0.000000001	0.000000000	0.000000003
0.000000004	0.000000001	0.000000010	0.000000003	0.000000007	<u>0.000000022</u>	0.000000001	0.000000014
0.000000001	0.000000000	0.000000003	0.000000001	0.000000002	0.000000001	0.000000000	0.000000004
0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000024
0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000

Table 4.10:3rd Iteration to Find Viterbi Matrix

XI. So the Viterbi matrix after the 3rd recursion is shown in table 4.11.

0.001823620	0.000009114	0.000000289	0.000000017
0.000651293	0.000153095	0.000000765	0.000000024
0.004949831	0.000038251	0.000008992	0.000000045
0.001302586	0.000051114	0.000000395	0.000000093
0.003516984	0.000051114	0.000002006	0.000000032
0.001693364	0.000038251	0.000000556	0.000000022
0.000390776	0.000001953	0.000000062	0.000000004
0.006903708	0.000153095	0.000001093	0.000000063

Table 4.11:Viterbi Matrix-After 3rd Recursion

XII. And the Backtrack matrix after the 3rd recursion is shown in table 4.12.

5	2	1	2
5	0	0	0
5	1	1	1
5	2	2	2
5	3	3	2
5	4	4	4
5	2	1	2
5	6	1	2

Table 4.12:Backtrack Matrix after 3rd Recursion

XIII. So for backtracking purpose, the greatest value at the 4th column of the final Viterbi matrix has to be found as shown in table 4.13.

0.001823620	0.000009114	0.000000289	0.000000017
0.000651293	0.000153095	0.000000765	0.000000024
0.004949831	0.000038251	0.000008992	0.000000045
0.001302586	0.000051114	0.000000395	<u>0.000000093</u>
0.003516984	0.000051114	0.000002006	0.000000032
0.001693364	0.000038251	0.000000556	0.000000022
0.000390776	0.000001953	0.000000062	0.000000004
0.006903708	0.000153095	0.000001093	0.000000063

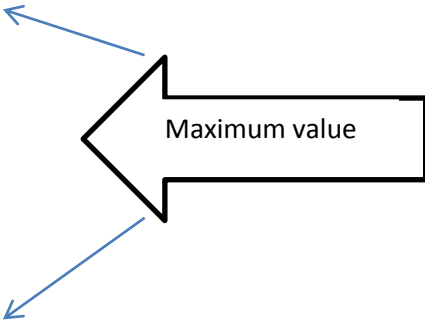


Table 4.13:Final Viterbi Matrix

XIV. So this work backtrack from the highest value using backtrack matrix

←	5	2	1	2	
	5	↖	0	0	
	5		←	1	
	5			←	2
	5				2
	5				4
	5				4
	5				2
	5				2

Table 4.14:Final Backtrack Matrix

XV. So the final predicted text fragment will be :

<s> नेपाल समूह 'ए' को अन्तिम

4.4 SRILM

SRILM[28, 29] is a toolkit for building and evaluating statistical language models (LMs). Most of the LM types it supports are based on N-gram statistics, including the standard back-off models, with an array of standard smoothing algorithms (Good-Turing, Witten-Bell, Kneser-Ney, etc.). Models based on word classes (automatically induced or externally defined) are also supported. SRILM implements various methods for interpolating and adapting LMs, as well as pruning for trading off size against performance. Once an LM is trained, it can be evaluated or used in a variety of standard tasks, such as perplexity computation, N-best and lattice rescoring, and text tagging and segmentation.

SRILM consists of three software layers. The core functionality is implemented in C++, comprising classes for containers (arrays, associative maps, tries, N-gram counts), application-related data structures (N-best lists, lattices, confusion networks, text statistics), smoothing methods, and the LMs themselves. The latter are arranged in a class hierarchy, reflecting the fact that many LM types are variants of more basic types, for which most of the implementation can be shared. The second layer, and the one most relevant to most users, is a set of executable tools that carry out the standard tasks of LM building and application, as well as manipulation of lattices, N-best lists, and confusion networks. A third layer comprises tools implemented as scripts in the Bourne shell, Gawk, and Perl languages for miscellaneous tasks that are carried out primarily through simple manipulation of text files or on top of the second-layer tools. The tools in the second and third layers are designed to combine via Unix pipes to carry out more complex tasks[29].

The techniques that help accommodate large data and models can be summarized as follows:

Count-of-count Statistics:

When count cutoffs on N-grams are used, as they usually are in training large LMs, it is not necessary to load all N-gram counts into memory. Rather, the count-of-count statistics required for discounting algorithms can be computed off-line (in constant memory), and for N-grams that fall below the frequency threshold only these count-of- need to be passed to the

LM estimation algorithm. This method is encapsulated in the SRILM `make-big-lm` script.

Vocabulary sub setting:

At test time, the data can usually be split into manageable chunks, each of which utilizes only a small subset of the vocabulary. All LM types now support a loading method triggered by N-gram `-limit-vocab`, whereby only those LM parameters are loaded that intersect with a given vocabulary subset.

Binary File Formats:

N-gram counts and backoff LMs now support binary formats that are inherently faster to read and, more importantly, support very fast loading with `-limit-vocab`, efficiently skipping over file portions that are outside a given vocabulary. While the format is binary, it is still portable between different byte-orders and machine word sizes.

Indirect Integer Storage:

Large counts require 64 bits for encoding, yet this would entail wasted space for most N-grams, which are still infrequent. An integer data type can be implemented that stores small count values inline using 2- or 4- byte words, while referencing a table for large values that do not fit in 15 or 31 bits, respectively.

Optimized Heap Allocation:

The overall overhead in heap memory allocation is lowered by providing a special-purpose allocator that keeps lists of small memory chunks, as are typically found in N-gram trie data structures. (This, too, exploits the fact that N-gram frequency distributions have a long tail of infrequent N-gram types, leading to a large number of trie nodes with small number of children.) For a 6-gram LM with 690M N-grams, this reduced memory usage by 30% (from 33GB to 23GB).

Destructive LM Merging:

Since large LMs are typically constructed as a static interpolation of component LMs by N-gram merging, N-gram LM merging is made destructive, so that the total memory demand is only the sum of the result LM and the smaller of the two inputs. This way, very large LMs can be built by successively merging the component models.

This work uses this SRILM toolkit to find the alpha and discounted value for the unigram and also alpha and discounted value for the bigram and the discounted probability for the trigram model. The snapshot of the output of the unigram, bigram and trigram model is given in figure 4.4, 4.5 and 4.6 respectively.

```

-3.448861  यहा -0.09469852
-3.749891  यहाँको -0.15083
-3.749891  यहाँस्थित -0.15083
-3.27277  यात्रा -0.2157018
-3.749891  यात्राको -0.1509072
-3.448861  यात्राबाट -0.09454393
-3.147831  यी -0.1965949
-3.27277  यू-१६ -0.1467149
-3.749891  यू-१७ -0.1507527
-3.448861  यू-१९ -0.1505208
-3.749891  यूक्रेन -0.1497467
-3.050921  यूक्रेनले -0.1045859
-3.448861  यूक्रेनविरुद्ध -0.1504434
-3.27277  यूक्रेनी -0.1505981
-3.448861  यूथ -0.15083
-3.749891  यूनाइटेडका -0.1506754
-3.749891  यूनाइटेडलाई -0.1507527
-3.448861  युनिभर्सलका -0.09469852
-2.635947  यूरो -0.5183965
-3.749891  यूरोकपको -0.1500565
-3.749891  यूरोप -0.15083
-3.749891  यूरोपा -0.15083
-2.708498  यूरोपियन -0.4358803
-3.147831  यूरोपेली -0.1196112
-3.749891  यूरोमा -0.1503661
-3.448861  यूवराज -0.09454393
-2.904793  यूवा -0.1289083

```

Figure 4.4: Unigram Model

```

-0.7072967  पनेरुले बनाएको 0.01903957
-1.185118  पराजित गरे 0.009272065
-0.841357  पराजित गरेको 0.1434708
-1.185118  पराजित गर्‍यो 0.5971459
-0.883388  परिषद् र -0.02394617
-0.4069666  परेको छ 1.193132
-0.7072967  पने ठाउँमा 0.06905053
-1.786478  पहिलो दुई -0.01944439
-1.360509  पहिलोपल्ट उपाधि 0.0189635
-1.360509  पहिलोपल्ट यूरोपियन 0.02327158
-0.883388  पाएका हुन् 1.204646
-0.883388  पाएको हो 1.034895
-0.8329354  पार गरेका 0.159061
-1.309357  पार गर्ने -0.01249145
-1.309357  प्रगेका छन् 1.706591
-0.4891745  प्रगेका थिए 1.342025
-0.7072967  पुलचोक इन्जिनियरिङका 0.06905053
-0.883388  पूरा आराम 0.06905053
-0.5319054  पोल्यान्ड र -0.3515127
-1.52021  प्रतियोगिता गरी 0.009693081
-1.552395  प्रतियोगितामा १ -0.01719687
-0.7072967  प्रतिस्पर्धीको ५४औं 0.06905053
-0.7072967  प्रदर्शनमा ब्रिटनले 0.06905053
-0.883388  प्रमिलाको लक्ष्य 0.06905053
-1.008327  प्रमिलाले बताइन् 0.06905053
-1.008327  प्रमिलाले बताएअनुसार 0.06905053
-0.8329354  प्रयास गरिरहेका 0.01234429
-1.309357  प्रयास गरेको -0.01919798

```

Figure 4.5: Bigram Model

-1.23896	हराएका छ </s>
-1.23896	च्याम्पियनसिपको छनोट नै
-1.23896	देशले छनोट पार
-0.06694679	गरएका छन् </s>
-0.8107345	जितेका छन् </s>
-1.23896	पनि छन् </s>
-1.23896	पुगीका छन् </s>
-0.06694679	बताएका छन् </s>
-0.06694679	भएका छन् </s>
-0.2513809	भिड्दै छन् </s>
-0.8107345	मा छन् </s>
-1.23896	लिएका छन् </s>
-1.23896	भनेका छन्, तर
-1.23896	गरिरहेका छन्, ओलम्पिक खेल्ने
-0.2513809	मैदान छाडेका थिए
-1.23896	घन्टा छुँदा रासेद
-1.53999	सहभागिता जनाउन लखन
-1.23896	भएकाले जबरजस्ती दौडने
-1.23896	गरको जापानले होखरससँग
-1.23896	अब जापानसँग खेल्ने
-1.23896	निश्चित जितको खोजीमा
-1.23896	स्वर्ण जितिन् </s>
-0.2513809	उपाधि जित्ने अभियानमा
-1.23896	मैले जिद्दी गरेपछि
-1.23896	फेरि जीवनमा नआउनसक्ने
-1.23896	स्टाक जोगार्दै इगल
-1.23896	उत्तरकोरियाको झण्डा स्किनमा

Figure 4.6: Trigram Model

4.5 JAVA Programming Language

Java is a programming language originally developed by James Gosling at Sun Microsystems and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities than either C or C++. Java applications are typically compiled to byte-code (class file) that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is a general-purpose, concurrent, class-based, object-oriented language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another. Java is as of 2012 one of the most popular programming languages in use, particularly for client-server web applications, with a reported 10 million users.

The original and reference implementation Java compilers, virtual machines, and class libraries were developed by Sun from 1995. As of May 2007, in compliance with the specifications of the Java Community Process, Sun relicensed most of its Java technologies under the GNU General Public License. Others have also developed alternative

implementations of these Sun technologies, such as the GNU Compiler for Java and GNU Class path.

CHAPTER 5

TESTING AND ANALYSIS

5.1 Nepali Corpus Data Statistics

The Nepali corpus named concatout.txt which contains 5241 Unicode Nepali words is used for training and testing. The corpus is collected from the Kantipur and Nagarik newspaper. The concatout.txt corpus consist of the words like

महिला १ सय मिटरमा देवीमाया पनेरुले बनाएको १२.०२ सेकेन्डको राष्ट्रिय कीर्तिमान तोड्ने प्रमिलाको लक्ष्य छ। सरकार परिवर्तनसँगै राखेपको नेतृत्व पनि फेरिने परम्परा छ। एकीकृत माओवादीको भागमा युवा तथा खेलकुद मन्त्रालय परेपछि लामा सदस्यसचिवका नियुक्त भएको वर्ष दिन बितेको छ। उनी आएलगत्तै केही खेल संघ विघटन गरे भने करार र ज्यालाका कर्मचारी हटाउने निर्णय गरे।

Figure 5.1: Corpus Sample

5.2 Training and Testing Data

Training and testing data is prepared form the original corpus. Training data consists of the text from the concatout.txt and also testing data for the model is obtained from the original corpus. This thesis uses the 35 test case. The 35 test case with the goal sentence is shown in the table 5.1.

	INITIAL TEXT FRAGMENT(INPUT)	GOAL SENTENCE(TESTING DATA)
1	चेक तीन	चेक तीन अंकमा दोस्रोमा छ</S>
2	नम्बर एक	नम्बर एक खेलाडी विकास श्रेष्ठलाई हराएर
3	ब्राजिल र	ब्राजिल र होन्डुरसका बीच हुनेछ
4	दमक क्याम्पसका	दमक क्याम्पसका सन्दिप चौधरीलाई हराए </S>
5	निश्चित जितको	निश्चित जितको खोजीमा थियो</S>
6	आइतबार राति	आइतबार राति राजधानी फर्के </S>
7	नेपाल फुटबल	नेपाल फुटबल संघले जनाएको छ

8	इजिप्टले बेलारुसलाई	इजिप्टले बेलारुसलाई ३-१ ले पराजित गर्‍यो
9	युरो २०१२	युरो २०१२ मा खेल्नेछन् </S>
10	दुईवर्षअघि	दुई वर्षअघि फिफा वल्डकप नै जित्यो
11	<S>नेपाली	नेपाली राष्ट्रिय टोलीले भाग लिने भएको छ
12	राष्ट्रिय कीर्तिमान	राष्ट्रिय कीर्तिमान तोड्ने प्रमिलाको लक्ष्य छ
13	पुल्चोक इन्जिनियरिडका	पुल्चोक इन्जिनियरिडका आशिष आचार्यसँग बराबरी खेले
14	<S>उदघाटन	<S>उदघाटन नहुँदै सुरु भएका खेलहरूमध्ये
15	<S>आर्मी	<S>आर्मी अफिसर्स क्लबमा उनले तेस्रो दिन
16	पराजित गरेको	पराजित गरेको जापानले होन्डुरससँग गोलरहीत बराबरी खेल्‍यो
17	गलत झण्डा	गलत झण्डा देखाएको भन्दै उत्तर कोरियाली
18	<S>तीन	तीन सातापछि युरो २०१२ आइतबार टुंगिँदैछ
19	<S>पहिलो	<S>पहिलो लेगमा १-० ले विजय पाएको छ
20	सोभियत संघबाट	सोभियत संघबाट स्वतन्त्र भएयता युक्रेनले पहिलोपल्ट
21	१-० ले	१-० ले विजय पाएको छ </S>
22	दक्षिण एसियाली	दक्षिण एसियाली खेलकुद (साग) को उस्सुमा नेपालले २ स्वर्ण
23	भएका केही	भएका केही खेलहरूमध्ये महिला फुटबल खेलमा
24	तीन वर्षयता	तीन वर्षयता पहिलोपल्ट उपाधि जित्ने अभियानमा
25	जापानले होन्डुरससँग	जापानले होन्डुरससँग गोलरहीत बराबरी खेल्‍यो </S>
26	दुई खेलमा	दुई खेलमा प्राइमले अन्तिमसम्म उत्कृष्ट प्रदर्शन
27	नेपाली खेलाडी	नेपाली खेलाडीछनोट पार गरेका छन्
28	एक घण्टा	एक घण्टा मैदान छाडेका थिए</S>
29	दुई स्ट्रोकको	दुई स्ट्रोकको अग्रता लिएका छन्</S>
30	सहभागिता जनाउन	सहभागिता जनाउन नेपाली युवा टिम शुक्रबार त्यसतर्फ उडेको छ,
31	तीन ओभर	तीन ओभर ७५ स्कोर गरेका उनले
32	दक्षिण कोरियाको	दक्षिण कोरियाको देखाउन पुगेका थिए </S>
33	गोल गरेपछि	गोल गरेपछि डिफेन्डिङ च्याम्पियन स्पेनले

34	निशान ढुंगेललाई	निशान ढुंगेललाई पराजित गरे भने चौथोमा
35	इन्जेक्सन लगाएर	इन्जेक्सन लगाएर प्रयास गरिरहेकाछन् </S>

Table 5.1: Test Data

5.3 Input and Output

	INITIAL TEXT FRAGMENT	PREDICTION USING BIGRAM	PREDICTION USING TRIGRAM	GOAL SENTENCE
1	चेक तीन	सातापछि युरो २०१२ को	<u>अंकमा दोस्रोमा छ </S></u>	चेक तीन अंकमा दोस्रोमा छ</S>
2	नम्बर एक	घण्टा मैदान छाडेका थिए	स्पेनले आयरल्यान्डलाई युरो २०१२	नम्बर एक खेलाडी विकास श्रेष्ठलाई हराएर
3	ब्राजिल र	मलेसिया भिड्दै छन् </S>	<u>होन्डुरसका बीच हुनेछ </S></u>	ब्राजिल र होन्डुरसका बीच हुनेछ</S>
4	दमक क्याम्पसका	<u>सन्दिप चौधरीलाई हराए </S></u>	<u>सन्दिप चौधरीलाई हराए </S></u>	सन्दिप चौधरीलाई हराए </S>
5	निश्चित जितको	<u>खोजीमा</u> भएको छ </S>	पराजित गरेको छ </S>	निश्चित जितको खोजीमा थियो</S>
6	आइतबार राति	ओलम्पिकको पुर्वसन्ध्यामा भएको छ	ओलम्पिकको तयारी गरेका छन्	आइतबार राति राजधानी फर्क </S>
7	नेपाल फुटबल	<u>संघले जनाएको छ </S></u>	<u>संघले जनाएको छ </S></u>	नेपाल फुटबल संघले जनाएको छ </S>
8	इजिप्टले बेलारुसलाई	<u>३-१ ले पराजित</u> गरेको	<u>३-१ ले</u> हरायो </S>	इजिप्टले बेलारुसलाई ३-१ ले पराजित गर्‍यो
9	युरो २०१२	<u>मा</u> भएको छ </S>	<u>मा</u> भएको छ </S>	युरो २०१२ मा खेल्नेछन् </S>
10	दुई वर्षअघि	<u>फिफा</u> वरीयता हेर्ने हो	<u>फिफा वल्डकप</u> जितेका छन्	दुई वर्षअघि फिफा वल्डकप नै जित्यो
11	<S> नेपाली	<u>टोली</u> भिड्दै छन् </S>	<u>टोली</u> सोमबार युरो २०१२	<S> नेपाली राष्ट्रिय टोलीले भाग लिने भएको छ
12	राष्ट्रिय कीर्तिमान	खेलाडीमा चुनेको थियो </S>	खेलाडीमा चुनेको थियो </S>	राष्ट्रिय कीर्तिमान तोड्ने प्रमिलाको लक्ष्य छ
13	पुल्चोक इन्जिनियरिडका	<u>आशिष आचार्यसँग बराबरी</u> </S>	<u>आशिष आचार्यसँग बराबरी</u> गोल	पुल्चोक इन्जिनियरिडका आशिष आचार्यसँग बराबरी खेले

14	<S>उदघाटन	<u>नहुँदै सुरु भएका</u> छन्	<u>नहुँदै सुरु भएका</u> छन्	<S>उदघाटन नहुँदै सुरु भएका खेलहरूमध्ये
15	<S>आर्मी	<u>अफिसर्स क्लबमा</u> सर्वाधिक सफल	<u>अफिसर्स क्लबमा</u> सर्वाधिक सफल	<S>आर्मी अफिसर्स क्लबमा उनले तेस्रो दिन
16	पराजित गरेको	रुसलाई १-० ले पराजित	बेलायतले १९०८ मा छन्	पराजित गरेको जापानले होन्डुरससँग गोलरहीत बराबरी खेल्यो
17	गलत झण्डा	स्क्रिनमा देखाउनु पर्ने ठाउँमा	स्क्रिनमा युरो २०१२ को	गलत झण्डा देखाएको भन्दै उत्तर कोरियाली
18	<S>तीन	<u>सातापछि युरो २०१२</u> को	अंकमा दोस्रोमा छ </S>	तीन सातापछि युरो २०१२ आइतबार टुंगिँदैछ
19	<S>पहिलो	<u>लेगमा १-० ले</u> पराजित	<u>लेगमा १-० ले</u> पराजित	<S>पहिलो लेगमा १-० ले विजय पाएको छ
20	सोभियत संघबाट	<u>स्वतन्त्र भएयता युक्रेनले</u> <u>पहिलोपल्ट</u>	<u>स्वतन्त्र भएयता युक्रेनले</u> </S>	सोभियत संघबाट स्वतन्त्र भएयता युक्रेनले पहिलोपल्ट
21	१-० ले	पराजित गरेको छ </S>	पराजित गरेको छ </S>	१-० ले विजय पाएको छ </S>
22	दक्षिण एसियाली	<u>खेलकुद</u> परिषद्का सदस्य सचिव	<u>खेलकुद</u> परिषद् (राखेप) ले	दक्षिण एसियाली खेलकुद (साग) को उसुमा नेपालले २ स्वर्ण
23	भएका केही	दिनमै महिला राष्ट्रिय कीर्तिमान	दिनमै महिला राष्ट्रिय कीर्तिमान	भएका केही खेलहरूमध्ये महिला फुटबल खेलमा
24	तीन वर्षयता	<u>पहिलोपल्ट</u> युरो २०१२ को	<u>पहिलोपल्ट</u> 'इन्डोर' मा छन्	तीन वर्षयता पहिलोपल्ट उपाधि जित्ने अभियानमा
25	जापानले होन्डुरससँग	१-० ले पराजित गरेको	१-० ले हरायो </S>	जापानले होन्डुरससँग गोलरहीत बराबरी खेल्यो </S>
26	दुई खेलमा	गीसलाई हराएको छ </S>	<u>प्राइमले अन्तिमसम्म उत्कृष्ट</u> <u>प्रदर्शन</u>	दुई खेलमा प्राइमले अन्तिमसम्म उत्कृष्ट प्रदर्शन

27	नेपाली खेलाडी	<u>छनोट</u> भएका छन् </S>	<u>छनोट पार गरेका छन्</u>	नेपाली खेलाडीछनोट पार गरेका छन्
28	एक घण्टा	<u>मैदान छाडेका थिए</u> </S>	<u>मैदान छाडेका थिए</u> </S>	एक घण्टा मैदान छाडेका थिए</S>
29	दुई स्ट्रोकको	<u>अग्रता लिएका छन्</u> </S>	<u>अग्रता लिएका छन्</u> </S>	दुई स्ट्रोकको अग्रता लिएका छन्</S>
30	सहभागिता जनाउन	लन्डन पुगेका थिए </S>	<u>नेपाली</u> टोली भिड्दै छन्	सहभागिता जनाउन नेपाली युवा टिम शुक्रबार त्यसतर्फ उडेको छ,
31	तीन ओभर	<u>७५ स्कोर</u> गरे </S>	<u>७५ स्कोर गरेका</u> छन्	तीन ओभर ७५ स्कोर गरेका उनले
32	दक्षिण कोरियाको	<u>देखाउन पुगेका थिए</u> </S>	<u>देखाउन पुगेका थिए</u> </S>	दक्षिण कोरियाको देखाउन पुगेका थिए </S>
33	गोल गरेपछि	इन्जेक्सन लगाएर प्रयास गरिरहेका	<u>डिफेन्डिड च्याम्पियन</u> भएका छन्	गोल गरेपछि डिफेन्डिड च्याम्पियन स्पेनले
34	निशान टुंगेललाई	<u>पराजित</u> गरेको छ </S>	<u>पराजित</u> गरेको छ </S>	निशान टुंगेललाई पराजित गरे भने चौथोमा
35	इन्जेक्सन लगाएर	<u>प्रयास</u> गरेको छ </S>	<u>प्रयास गरिरहेका</u> २ सय	इन्जेक्सन लगाएर प्रयास गरिरहेका छन् </S>

Table 5.2: Input and Output of the Prediction Model

5.3.1 Snapshot of the output of the trigram and bigram model

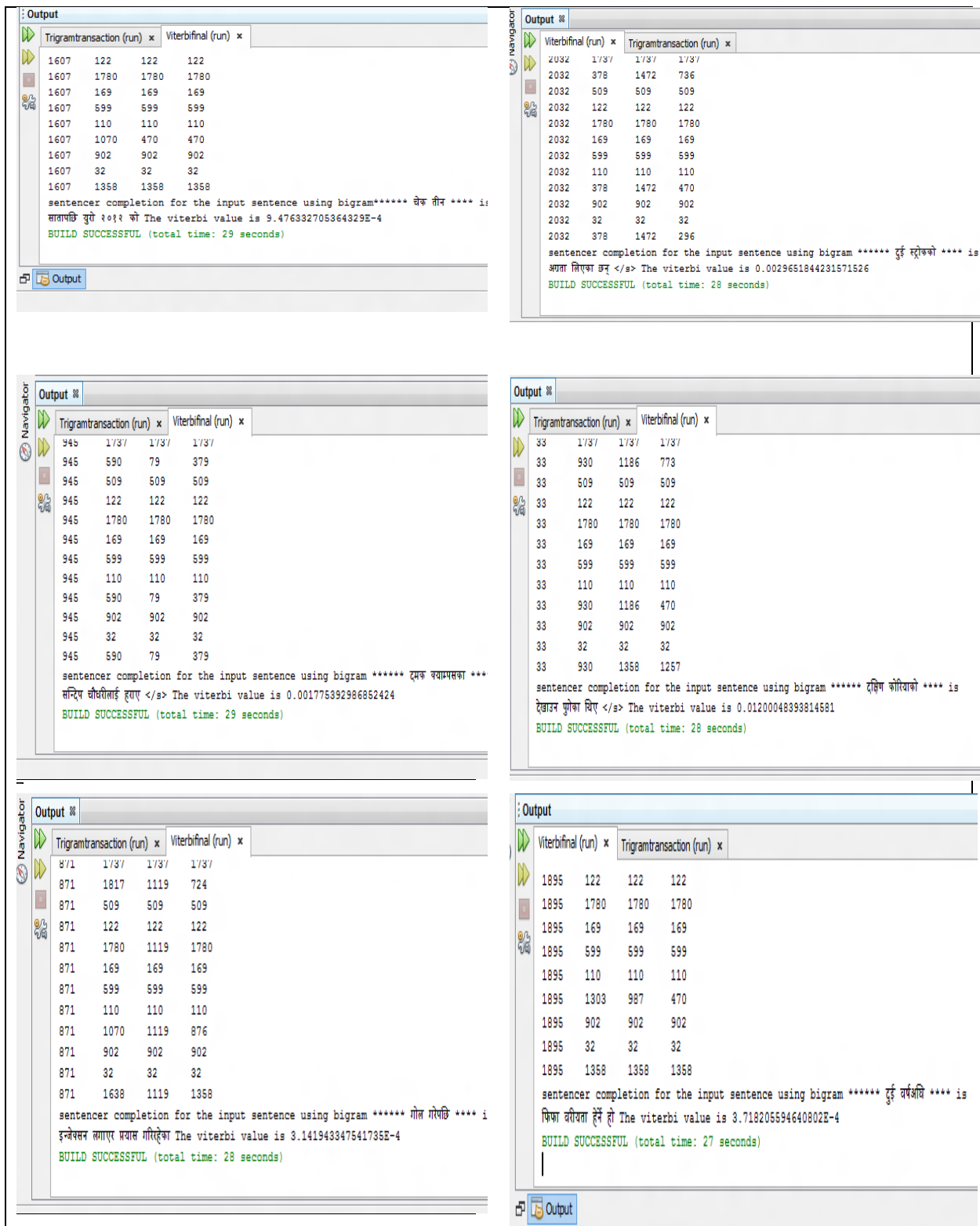


Figure 5.2: Snapshot of Bigram Output

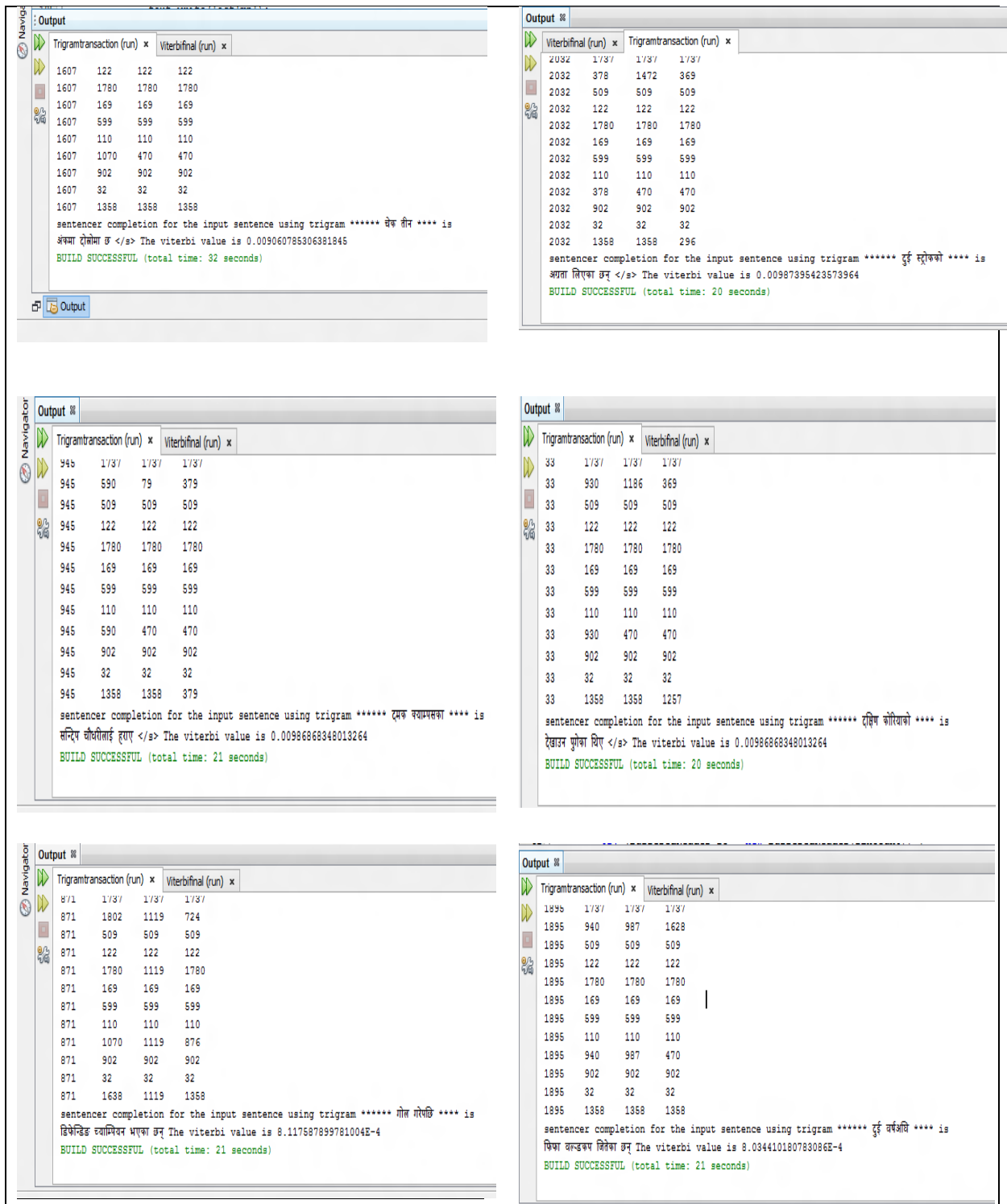


Figure 5.3: Snapshot of Trigram Output

5.4 Analysis of the Prediction Model

Given an initial text fragment, a predictor that solves the sentence completion problem has to conjecture as much of the sentence that the user currently intends to write but preferably, but not necessarily, the entire remainder[13].

What is an appropriate performance measure for this problem? The perceived benefit of an assistance system is highly subjective, but it is influenced by measurable quantitative factors. So, a system of three performance indicators is defined.

For the evaluation this thesis will use precision, recall and F-Score. Recall equals the fraction of saved keystrokes. Precision is the ratio of characters that the users have to scan for each character they accept. F_1 score (also F-score or F-measure) is a measure of a test's accuracy. It considers both the precision and the recall of the test to compute the score.

$$\text{Precision} = \frac{\sum \text{Accepted completion of words}}{\sum \text{Suggested completion of word}} \quad (5.1)$$

$$\text{Recall} = \frac{\sum \text{Accepted completion of words}}{\sum \text{Length of missing part for sentence completion}} \quad (5.2)$$

$$\text{F-Score} = 2 * \frac{\text{Precision*Recall}}{\text{Precision+Recall}} \quad (5.3)$$

5.4.1 Calculation of Precision, Recall And F-Score of Bigram Prediction Model

$$\text{Precision} = \frac{52}{140} = 0.371429$$

$$\text{Recall} = \frac{52}{88} = 0.590909091$$

$$\text{F-Score} = 2 * \frac{0.371429 * 0.590909091}{0.371429 + 0.590909091} = 0.456140351$$

5.4.2 Calculation of Precision, Recall and F-Score of Tri-gram Prediction Model

$$\text{Precision} = \frac{67}{140} = 0.47857$$

$$\text{Recall} = \frac{67}{73} = 0.91780$$

$$\text{F-Score} = 2 * \frac{0.47857 * 0.91780}{0.47857 + 0.91780} = 0.62910$$

5.4.3 Comparison between Bigram and Trigram Prediction Model

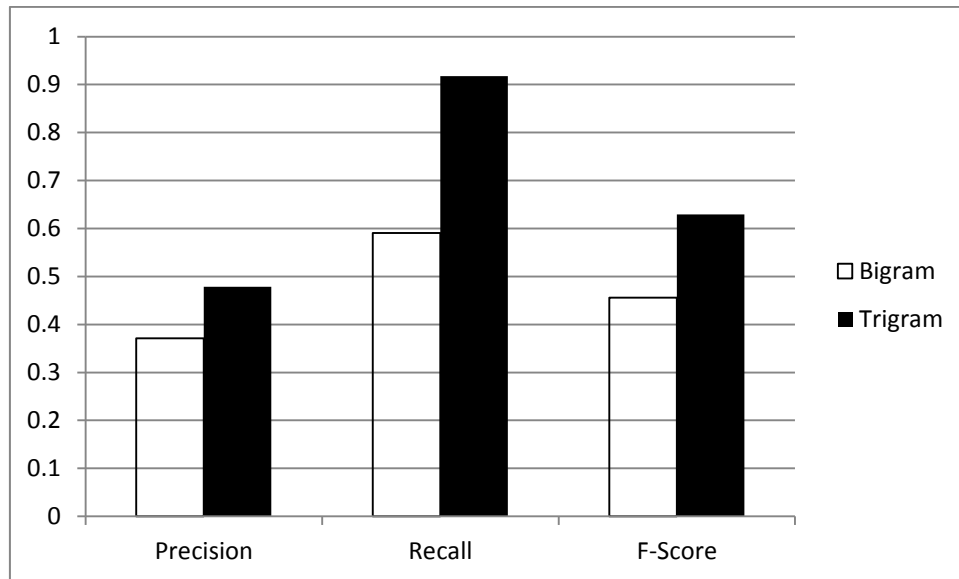


Figure 5.4: Comparison between Bigram and Trigram Prediction Model

5.5 Discussion of the Result

The result is found using the mechanism precision, recall and F-score. The F-score can be interpreted as a harmonic mean of the precision and recall, where an F-score reaches its best value at 1 and worst score at 0. Thus, by analyzing the result for all the 35 input, Trigram Prediction Model is 62.9% accuracy whereas Bigram Prediction Model is 45.6% accuracy. From this finding it is clear that overall Trigram Prediction Model is more accurate than Bigram Prediction Model.

But for Nepali language, the output for some of the input to the Bigram Prediction Model is better than the Trigram Prediction Model. For the input cases 5, 8, 18, 20 the Bigram Prediction Model is better than the Trigram Prediction Model. Also there are cases 4, 7, 13, 14, 28, 29, 32 where both the Trigram and Bigram Prediction Model have the same output.

There are also input cases where both Bigram Prediction Model and Trigram Prediction Model prediction sentence has not been selected as in test case 2,6,12,17,23,25.

CHAPTER 6

CONCLUSION AND RECOMMENDATION

6.1 Conclusion

This thesis has predicted the sentence fragment for sentence completion with input text fragment using Bigram and Trigram Language Model with Viterbi algorithm as a decoding algorithm which is a dynamic algorithm. By using the measuring criteria Precision, Recall and F-score, this thesis found out that Trigram Language Model predicts the word more accurately than Bigram Language Model. It is found that overall 62.9% accuracy of Trigram Language Model whereas 45.6% for Bigram Language model. But for individual input there are cases where some of the output of the Bigram Prediction Model is better than Trigram Prediction Model. Also there are some input cases where the output of the Bigram and Trigram Prediction Model is not selected.

Prediction using Trigram Prediction Model has more complexity than bigram prediction model in terms of time. This work used the corpus consisting of 5241 Unicode Nepali words which is used for training and testing. Due to the limitation of the size of the words in the corpus, the no of bigram and trigram counts is low. When this work predicts the word segment based on the bigram and trigram counts, results in low accuracy. So to increase the accuracy, the corpus size should be increased.

6.2 Recommendation

This work has predicted the sentence fragment using Bigram and Trigram Language Model. There are some cases where both the Bigram and Trigram Prediction Model output is not selected by the use, so further 4-gram, 5-gram, up to N-gram can be used to predict the sentence may solve the problem. By increasing the value of N, the accuracy of the system may increase but also increase the complexity in terms of time. Not only the sentence completion can be analyzed, but also the grammar judgment of the sentence completion task can be performed. Also the size of corpus could be increased to get the better result. Furthermore, Confabulation Based method, Graph Method can be used to complete the sentence.

References

- [1] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*: Pearson Education, 2003.
- [2] E. Reiter and R. Dale, *Building Natural Language Generation Systems*: Cambridge University Press, 2000.
- [3] S. Chaudhuri and R. Kaushik, "Extending Autocompletion To Tolerate Errors," *Proceedings of the 35th SIGMOD international conference on Management of data - SIGMOD '09*, pp. 707-718, 2009.
- [4] K. Trnka and K. F. McCoy, "Corpus studies in word prediction," *Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility - Assets '07*, october 2007.
- [5] E. D. Liddy, "Natural Language Processing," in *Encyclopedia of Library and Information Science*, 2nd ed: Marcel Decker, 2001.
- [6] (2012, September 22). <http://en.wikipedia.org/wiki/Phonology>.
- [7] J. Zhou, "Educational Tool For Charnik's Marker-Passing Algorithm," Master of Science in Computer Science, San Diego State University, 2011.
- [8] B. Manaris, "Natural Language Processing: A Human–Computer Interaction Perspective," *Advances in Computers (Marvin V. Zelkowitz, ed.)*, vol. 47, p. 66, 1998.
- [9] (September 22). <http://omarsbrain.wordpress.com/2010/08/12/natural-language-processing-the-big-picture/>.
- [10] R. Rosenfield, "Two Decades of Statistical Language Modeling- Where Do We Go From," *Computer Science Department, Carnegie Mellon University*, vol. 1321, 2000.
- [11] Z. Wei, D. Miao, J.-H. Chauchat, R. Zhao, and W. Li, "N-grams based feature selection and text representation for Chinese Text Classification," *International Journal of Computational Intelligence Systems*, vol. Vol.2, pp. 365-374, December, 2009.
- [12] N. Agrawal and M. Swain, "Auto Complete Using Graph Mining: A Different Approach " *IEEE*, 2011.
- [13] S. Bickel, P. Haider, and T. Scheffer, "Predicting sentences using N-gram language models," *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pp. 193-200, 2005.

- [14] L. Nepveu, G. Lapalme, P. Langlais, and G. Foster, "Adaptive Language and Translation Models for Interactive Machine Translation," *Conference on Empirical Methods on Natural Language Processing*, 2005.
- [15] C. E. Shannon, "Prediction and Entropy of Printed English," *In Bell systems Technical Journal*, vol. 30, pp. 50-60, 1948.
- [16] J. Darragh and I. Witten, *The Reactive Keyboard*: Cambridge University Press, 1992.
- [17] H. Motoda and K. Yoshida, "Machine Learning Techniques to Make Computers Easier to Use," *In proceedings of the fifteenth International Joint Conference on Artificial Intelligence*, 1997.
- [18] B. D. Davison and H. Hirsh, "Predicting Sequences of User Actions," *AAAI/ICML Workshop on Predicting the Future: AI Approaches to Time-Series Analysis*, 1998
- [19] B. Korvemaker and R. Greiner, "Predicting UNIX Command Lines : Adjusting to User Patterns," *American Association for Artificial Intelligence*, 2000.
- [20] K. Grabski and T. Scheffer, "Sentence Completion," *Proceedings of the 27th annual international ACM SIGIR conference on Research and Development in Information Retrieval*, 2004.
- [21] Q. Qiu, Q. Wu, D. J. Burns, M. J. Moore, R. E. Pino, M. Bishop, and R. W. Linderman, "Confabulation Based Sentence Completion for Machine Reading," *IEEE*, 2011.
- [22] A. Renaud, F. Shein, and V. Tsang, "Grammaticality Judgement in a Word Completion Task," *Proceedings of the NAACL HLT 2010 Workshop on Computational Linguistics and Writing*, pp. 15-23, 2010.
- [23] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*: MIT Press, 1999.
- [24] D. Jurafsky and J. H. Martin, *An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*: Draft of February 11, 2007, 2006.
- [25] A. Almeida and P. Bhattacharyya, "Experiments in N-gram based indexing and retrieval in Marathi ".
- [26] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai, "Class-Based N-gram Models of Natural Language," *Association for computer linguistics*,, 1992.

- [27] R. Sramek, "The on-line Viterbi algorithm," Masters, Department of Computer Science Faculty of Mathematics, Physics and Informatics Comenius University, Bratislava, 2007.
- [28] A. Stolcke, "SRILM — An Extensible Language Modeling Toolkit," *Proc. Intl. Conf. on Spoken Language Processing*, vol. 2, p. 4, 2002.
- [29] A. Stolcke, J. Zheng, W. Wang, and V. Abrash, "SRILM at Sixteen: Update and Outlook."

Appendix

```
// program to create the bigram transition matrix

package bigram;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.util.Hashtable;

public class Bigram {

    public static Hashtable<String, Integer> h=new Hashtable<>();
    public static Hashtable<Integer, String> ih=new Hashtable<>();
    public static Hashtable<String,UnigramBackoff> UH=new Hashtable<>();

    public static void main(String[] args) throws Exception {

        Bigram b= new Bigram();
        b.MakeHash();
        // b.MakeCountMatrix();
        b.UnigramHashTable();
        b.BigramProbability();
    }

    // function to make the hash table for the entire vocab and its inverse also

    public void MakeHash() throws Exception
    {
        FileReader fin=new FileReader("vocab3.txt");
        String s;
        try (BufferedReader f = new BufferedReader(fin)) {
            int i=1;
            h.put("<s>", 0);
            ih.put(0, "<s>");

            while((s=f.readLine())!=null)
            {
                h.put(s, i);
                ih.put(i, s);
                i++;
            }
            h.put("</s>",i);
            ih.put(i, "</s>");
        }
    }

    //function to make the bigram count matrix table
```

```

public void MakeCountMatrix() throws Exception {
    int countmatrix[][]=new int[h.size()][h.size()]; //initialization the size of the count matrix
    String s;
    FileReader fincount=new FileReader("countout.txt");
    try (BufferedReader fc = new BufferedReader(fincount)) {
        while((s=fc.readLine())!=null)
        {
            int j,k;
            String []tokens=s.split("\t");
            String []subtokens=tokens[0].split(" ");
            j=h.get(subtokens[0]);
            k=h.get(subtokens[1]);
            System.out.println(j+k);
            System.out.println(subtokens[0]+\t"+subtokens[1]);
            countmatrix[j][k]= Integer.parseInt(tokens[1]);
        }
    }
    for(int l=0;l<h.size();l++)
    {
        for(int m=0;m<h.size();m++) {
            System.out.print(countmatrix[l][m]+\t");
        }
        System.out.println();
    }
}
// to make the unigramhash table containing the discounted probability and also the alpha value

public void UnigramHashTable()throws Exception{
    String s;
    FileReader fincount=new FileReader("probability13.txt"); //read the file contaning the unigram
discounted probability and the alpha value
    try (BufferedReader fc = new BufferedReader(fincount))
    {
        while((s=fc.readLine())!=null){
            String []token=s.split("\t");
            UnigramBackoff b=new UnigramBackoff(Double.valueOf(token[0].trim()).doubleValue(),
Double.valueOf(token[2].trim()).doubleValue());
            UH.put(token[1],b);
        }

        fc.close();

    }catch(Exception e)
    {
        System.out.println(e);
    }
    System.out.println(UH);
}

// to make the bigram probability matrix

public void BigramProbability() throws Exception{
    String s;

```

```

double BigramMatrix[][]=new double[h.size()][h.size()];
UnigramBackoff b1=new UnigramBackoff();
UnigramBackoff b2=new UnigramBackoff();
FileReader fincount=new FileReader("probability23.txt"); // to read the file containing the
discounted probability of the bigram words

```

```

try (BufferedReader fc = new BufferedReader(fincount)) {

    // to input the discounted probability in the bigram matrix

    while((s=fc.readLine())!=null)
    {

        int j,k;
        String []tokens=s.split("\t");
        String []subtokens=tokens[1].split(" ");
        j=h.get(subtokens[0]);
        k=h.get(subtokens[1]);
        BigramMatrix[j][k]= Double.parseDouble((tokens[0]));
    }
    fc.close();
}catch(Exception e)

{
    System.out.println(e);
}

// to print the matrix before backoff

System.out.println("Before Backoff");
for(int l=0;l<h.size();l++){
    for(int m=0;m<h.size();m++) {
        System.out.print(BigramMatrix[l][m)+"\t");
    }
    System.out.println();
}

// loop for the backoff purpose

for(int l=0;l<h.size();l++)
{
    for(int m=0;m<h.size();m++)
    {
        if(BigramMatrix[l][m]==0.0){
            String s1=ih.get(l);
            String s2=ih.get(m);
            b1=UH.get(s1);
            b2=UH.get(s2);
            BigramMatrix[l][m]=b1.getAlphavalue()+b2.getDiscountedProb();
        }
    }
}

```

```

    }

    // writing the bigrammatrix in the file ....to use the viterbi in that bigram matrix

    FileWriter fw=new FileWriter("testaftercommenting.txt");
    BufferedWriter fout=new BufferedWriter(fw);

    for(int l=0;l<h.size();l++)
    {
        for(int m=0;m<h.size();m++)

            {
                fout.write(BigramMatrix[l][m)+"\t");
            }
        fout.write("\n");
    }
}
}

```

```
package bigram;
```

```

public class UnigramBackoff {
    private double discountedProb;
    private double alphavalue;

    public UnigramBackoff(){
        discountedProb=0.0;
        alphavalue=0.0;
    }
    public UnigramBackoff(double a){
        discountedProb=a;
    }
    // to assing the value to discountedprob and the alpha value which receives as an argument
    public UnigramBackoff(double a, double b){
        discountedProb=a;
        alphavalue=b;
    }
    public double getDiscountedProb(){
        return discountedProb;
    }
    public double getAlphavalue(){
        return alphavalue;
    }
}

```

```
//program to use the Viterbi algorithm to decode the sequence in bigram matrix transition
```

```
package viterbifinal;
```

```

import java.io.BufferedReader;
import java.io.FileReader;
import java.util.Hashtable;

```

```

import java.math.*;

public class Viterbifinal {

public static Hashtable<String, Integer> h=new Hashtable<>();
public static Hashtable<Integer, String> ih=new Hashtable<>();

public static void main(String[] args) throws Exception

{

    FileReader fin=new FileReader("vocab.txt");
    String s;
    try (BufferedReader fc = new BufferedReader(fin)) {
        int k=0;
        while((s=fc.readLine())!=null)
        {
            h.put(s, k);
            ih.put(k, s);
            k++;
        }

        // to read the bigram transition table from the text file

        FileReader fi=new FileReader("transition.txt");
        double Tmatrix[][]=new double[h.size()][h.size()];
        try (BufferedReader f = new BufferedReader(fi)){
            int i=0;
            while((s=f.readLine())!=null){
                String []tokens=s.split("\t");
                for(int j=0;j<tokens.length;j++)
                {
                    Tmatrix[i][j]=Double.parseDouble(tokens[j]);
                    Tmatrix[i][j]=Math.pow(10.0, Double.parseDouble(tokens[j]));
                }
                i++;
            }
            double [][]Vmatrix=new double[h.size()][4];
            int [][]Bmatrix= new int[h.size()][4];

// System.out.println("Enter the fragment of text");

// to read the input from the text file

FileReader finput=new FileReader("input.txt");
BufferedReader inputbuffer = new BufferedReader(finput);
int l=0;
// String inputtextfragment;
// inputtextfragment=inputbuffer.readLine();

// to read the last word of the input sentence for the Sentence Completion Purpose

```

```

while((s=inputbuffer.readLine())!=null){
    String []tokensinput=s.split(" ");
    System.out.println(tokensinput[tokensinput.length-1]);
    l=h.get(tokensinput[tokensinput.length-1]);
}

//Veterbi Matrix Intialization

System.out.println("vmatrix initializlnation print");

for(int j=0;j<h.size();j++)
{
    Vmatrix[j][0]=Tmatrix[l][j];
//    System.out.println(Vmatrix[j][0)+"\t");
}

//Backtrack Matrix Initialization

System.out.println("backtrack initialization print");

for(int j=0;j<h.size();j++)
{
    Bmatrix[j][0]=l;
//    System.out.println(Bmatrix[j][0)+"\t");
}

// viterbi matrix and backtrack matrix calculation

int index=0;

for(int t=1;t<4;t++) //for calculating veterbi matrix columns
{
    for(int ip=0;ip<h.size();ip++) //loop for searching max value
    {
        double value=0.0;

        for(int j=0;j<h.size();j++)
        {
            if(Vmatrix[j][t-1]*Tmatrix[j][ip]>value)
            {
                value=Vmatrix[j][t-1]*Tmatrix[j][ip];
                index=j;
            }
        }

        Vmatrix[ip][t]=value;
        Bmatrix[ip][t]=index;
    }
}

```



```

//printing the Veterbi Matrix

System.out.println("The verterbi matrix is");
for(i=0;i<h.size();i++)
{
for(int j=0;j<4;j++)
{
System.out.print(Vmatrix[i][j)+"\t");

}
System.out.println();
}

//printing the Backtrack Matrix

System.out.println("The Backtrack matrix");
for(i=0;i<h.size();i++)
{
for(int j=0;j<4;j++)
{

System.out.print(Bmatrix[i][j)+"\t");
}
System.out.println();
}

//to calculate the greatest value at the last row of the viterbi matrix

int tbindex=0; //traceback index

double greatestvalue =0.0; //to find the maximum no of the viterbi matrix

for(int ak =0;ak<h.size();ak++)
{
if(Vmatrix[ak][3]>greatestvalue)
{
greatestvalue= Vmatrix[ak][3];
tbindex=ak;
}
}

//to find the sequence for the sentence completion by bactracking

String []last = new String [5];
last[4]=ih.get(tbindex);
int a;
for(int ij=3;ij>0;ij--)
{
a=Bmatrix[tbindex][ij];
last[ij]=ih.get(a);
tbindex=a;
}

```

```

// reading input from the file for the printing purpose

    FileReader finputforsentencecompletion=new FileReader("input.txt");
    BufferedReader inputbufferforprintingpurpose = new
BufferedReader(finputforsentencecompletion);
    String inputforprinting;
    inputforprinting=inputbufferforprintingpurpose.readLine();
    System.out.println("sentencer completion for the input sentence***** " + inputforprinting + "
**** is ");

    // giving the output text fragment for the sentence completion
    for (int mn =1;mn<5;mn++)
    {
        System.out.print(last[mn] + " ");
    }

    System.out.println("The viterbi value is " + greatestvalue );

}
}
}
}

```

// program to create the trigram transition matrix and use of the Viterbi algorithm for sentence decoding

```
package trigramtransaction;
```

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.util.*;
import java.math.*;
```

```
public class Trigramtransaction {
```

```

    public static Hashtable<String, Integer> h=new Hashtable<>();
    public static Hashtable<Integer, String> ih=new Hashtable<>();
    public static Hashtable<String,Ubackoff> UH=new Hashtable<>();
    public static Hashtable<String,Bbackoff> BH=new Hashtable<>();
    public static Hashtable<Integer, String> ICV= new Hashtable<>();
    public static Hashtable<String, Integer> CV=new Hashtable<>();
    public static double Tmatrix[][]=new double[ih.size()][ih.size()];//initialization of the bigram
transition matrix
    public static double TrigramMatrix[][]=new double[CV.size()][h.size()];
    public static void main(String[] args) throws Exception {
```

```

        Trigramtransaction t= new Trigramtransaction();
        t.MakeHash();
```

```

Tmatrix= t.MakeBigram();
t.TrigramProbability();
TrigramMatrix=t.ValueTrigramProbability();
t.veterbi();
}

// to make the hash table of the vocabulary

public void MakeHash() throws Exception
{
FileReader fin=new FileReader("vocab.txt");
String s;
try (BufferedReader fc = new BufferedReader(fin)) {
int k=0;
while((s=fc.readLine())!=null)
{
h.put(s.trim(), k);
ih.put(k, s.trim());
k++;
}

// to read bigram transition table from file....for backoff purpose

}
}

public double[][] MakeBigram() throws Exception
{
double Tmatrix[][]=new double[ih.size()][ih.size()];
// reading the bigramtransition text file and assign it in Tmatrix
FileReader fi=new FileReader("transition.txt");
String s;
try (BufferedReader f = new BufferedReader(fi)){
int i=0;

while((s=f.readLine())!=null){
String []tokens=s.split("\t");
for(int j=0;j<tokens.length;j++)
{
Tmatrix[i][j]=Math.pow(10.0,Double.parseDouble(tokens[j]));

}
i++;
}
}
return Tmatrix;
}

// to make the hash table of the rows index of the trigram transtion matrix

public void TrigramProbability() throws Exception{
String s;
//reading the textfile containing the trigram discounted probability file

```

```

FileReader fincount=new FileReader("tirgramprobability.txt");
try (BufferedReader fc = new BufferedReader(fincount)) {
    int z=0;
    while((s=fc.readLine())!=null)
    {
        int j,k;
        String []tokens=s.split("\t");
        String []subtokens=tokens[1].split(" ");
        String cd=subtokens[0]+" "+subtokens[1];
        String ef=subtokens[2];

        ICV.put(z,cd);
        CV.put(cd, z);
        z++;

    }

    fc.close();
}catch(Exception e)

{
    System.out.println(e);
}

}

// to make the trigram transition matrix

public double[][] ValueTrigramProbability() throws Exception{
    double TrigramMatrix[][]=new double[CV.size()][h.size()];
    String s1;
    FileReader fincount=new FileReader("tirgramprobability.txt");
    try (BufferedReader fcc = new BufferedReader(fincount)) {

        while((s1=fcc.readLine())!=null)
        {
            // System.out.print("start");
            int j,k;
            String []tokens=s1.split("\t");
            String []subtokens=tokens[1].split(" ");
            String cd=subtokens[0]+" "+subtokens[1];
            String ef=subtokens[2];
            j=CV.get(cd);
            k=h.get(ef);
            if(j<CV.size() && k<h.size()){
                TrigramMatrix[j][k]=Math.pow(10.0,Double.parseDouble((tokens[0])));
            }
        }

        fcc.close();
    }catch(Exception e)

    {

```

```

        System.out.println(e);
    }

    for(int l=0;l<CV.size();l++)
    {
        for(int m=0;m<h.size();m++)
        {
            if(TrigramMatrix[l][m]==0.0){
                String str1=ICV.get(l);
                String str2=ih.get(m);
                // System.out.println(str1+" "+str2);
                String []tokens=str1.split(" ");
                int index1=h.get(tokens[1]);
                int index2=h.get(str2);

                TrigramMatrix[l][m]=Tmatrix[index1][index2];
            }
        }
    }

    return TrigramMatrix;
}

//function to apply the viterbi decoding algorithm to trigram transition table

public void veterbi() throws Exception
{
    double [][]Vmatrix=new double[h.size()][4];
    int [][]Bmatrix= new int[h.size()][4];

    FileReader finput=new FileReader("input.txt");
    BufferedReader inputbuffer = new BufferedReader(finput);
    // FileWriter fout1=new FileWriter("out");
    // BufferedWriter outbuffer=new BufferedWriter(fout1);
    String s;
    String []tokensinput=null;
    int l=0;
    while((s=inputbuffer.readLine())!=null){
        tokensinput=s.split(" ");
        System.out.println(tokensinput[tokensinput.length-1]);
        l=CV.get(tokensinput[tokensinput.length-2]+" "+tokensinput[tokensinput.length-1]);
        System.out.print("value of l is " + l);

    }

    System.out.print("value of l is " + l);
    //Veterbi Matrix Intialization
    System.out.println("Vmatrix initializlnation print");

    for(int j=0;j<h.size();j++)

```

```

    {
    Vmatrix[j][0]=TrigramMatrix[1][j];
    // System.out.println(Vmatrix[j][0)+"\t");
    }
    //Backtrack Matrix Initialization
    System.out.println("backtrack initialization print");
    int b=h.get(tokensinput[tokensinput.length-1]);
    for(int j=0;j<h.size();j++)
    {
    Bmatrix[j][0]=b;
    // System.out.println(Bmatrix[j][0)+"\t");
    }

```

// viterbi and backtrack matrix calculation

```

int index=0;

for(int t=1;t<4;t++) //for calculating veterbi matrix columns
{

for(int ip=0;ip<h.size();ip++) //loop for searching max value
{
double value=0.0;
if(Vmatrix[j][t-1]*Tmatrix[j][ip]>value)
{
value=Vmatrix[j][t-1]*Tmatrix[j][ip];
index=j;
}
}

Vmatrix[ip][t]=value;
Bmatrix[ip][t]=index;
}
}

```

// printing the Veterbi Matrix

```

System.out.println("The verterbi matrix is");
for(int i=0;i<h.size();i++)
{
for(int j=0;j<4;j++)
{
System.out.print(Vmatrix[i][j)+"\t");
}
}
System.out.println();
}

```

//printing the Backtrack Matrix

```

System.out.println("The Backtrack matrix");
for(int i=0;i<h.size();i++)

```

```

{
for(int j=0;j<4;j++)
{

    System.out.print(Bmatrix[i][j)+"\t");
}
System.out.println();
}

//to calculate the greatest value at the last row of the viterbi matrix

int tbindex=0;           //traceback index
double greatestvalue =0.0; //to find the maximum no of the viterbi matrix

for(int ak =0;ak<h.size();ak++)
{
    if(Vmatrix[ak][1]>greatestvalue)
    {
        greatestvalue= Vmatrix[ak][1];
        tbindex=ak;
    }
}

//to find the sequence for the sentence completion

String []last = new String [5];
last[4]=ih.get(tbindex);
// System.out.println(last);

int a;
for(int ij=3;ij>0;ij--)
{

    a=Bmatrix[tbindex][ij];
// System.out.println(a);
    last[ij]=ih.get(a);
    tbindex=a;
// System.out.println();

}
// reading input from the file for the printing purpose

FileReader finputforsentencecompletion=new FileReader("input.txt");
BufferedReader inputbufferforprintingpurpose = new
BufferedReader(finputforsentencecompletion);
String inputforprinting;
inputforprinting=inputbufferforprintingpurpose.readLine();
System.out.println("sentencer completion for the input sentence***** " + inputforprinting + "
**** is ");
for (int mn =1;mn<5;mn++)
{
    System.out.print(last[mn] + " ");
}

```

```
    System.out.println("The viterbi value is " + greatestvalue );  
  }  
}
```