Chapter 1

INTRODUCTION

1.1 Machine Translation

Machine Translation (MT) [10] is to translate a text from one natural language (i.e. source language) to another language (target language) using computer with a little or no human involvement; hence it is also known as Automatic Translation (AT).

Machine translation is one of the well known and most important applications of Natural Language Processing (NLP) and is as old as the history of the modern digital computing. It was first envisioned in 1950s and still is a most happening application in the field of NLP. MT has become a testing ground in computer science, linguistics and Artificial intelligence. It was first implemented by IBM in 1954 with a basic word for word translation system [2]. It expects that the text in some specific language as input (source language) produces an output text in any other language (target language) with the same semantics i.e. a translation is not said to be successful until the overall meaning of the text is not preserved in the process.

The scope of the machine translation is not developed so far that it ends the need of the human translator, but it can be used to assist them in different ways. It can be used to translate various technical papers and manuals where there is not used any symbolic language and fancy words that are not so common and their subject field and style are simple as well as inside some domain [1]. MT can also provide raw translation which can be revised or "post-edited" to give a high quality translation in a shorter time.

1.2 Approaches in Machine Translation

There are two approaches of machine translation one is rule based approach and another is data driven approach. In rule based approach different tools like parser and morphological analyzers [8] is used to generate some intermediate representation by analyzing the source language. Target language text is generated from this intermediate representation using different rules. These rules are specially used to transfer different grammatical structures from source language to target language.

The data driven approach uses parallel corpus[1] and monolingual corpus[2] for the translation purpose. We can use either statistical method or example based method in data driven approach. In statistical method target language is generated from source language on the basis of some statistical model, Baye's rule and statistical decision theory is used in this approach to solve the decision problem. Example based machine translation (EBMT) uses case based machine learning approach i.e. it uses solutions of past problems to solve new problem.

For the rule based system a large number of rules should be formulated which is a very complex process and the complexity increases as the rules grow large and it consumes a lot of time too. Since the structure of various languages do not resembles, the rules generated for one language pair may not work for different language pairs.

1.3 Statistical Machine Translation

Statistical Machine Translation (SMT) is the name for a class of approaches that do just the translation of text from one language to other, by building the probabilistic models of faithfulness[3] and fluency [9], and then combining these models to choose the most probable translation. It is the discipline in which translation system makes use of two parallel corpora. The framework of statistical machine translation formulates the problem of translating a sentence in a language E into another language N (here English as source and Nepali as target language) as the maximization problem of the conditional probability

$$\check{N} = \text{argmax}_N P(N|E)$$

The application of the Baye's Rule resulted in

$$\check{N} = \text{argmax}_N P(N)P(E|N).$$

The former term $P(N)$ is called a language model, representing the likelihood of N. The latter term $P(E|N)$ is called a translation model, representing the generation probability from N into E.

According to [10], a translation which is both very close in terms of the semantics to source language and natural as an utterance in the target language is unlikely to happen. If it is going to

[1] A text placed alongside its translation or translations.
[2] A corpus that contains texts in a single language.
[3] Tries to find the closest word in terms of its semantic.

be translated anyway it should be compromised. This is exactly what is done in statistical machine translation; the sentence in the target language which is most probable and somehow acceptable is produced i.e.

$$\text{Best translation } \check{T} = \text{argmax}_T \text{ faithfulness (T, S) fluency (T)}$$

This resembles the Bayesian model explained above.

1.4 Problems in Machine Translation

Translation is one of the very difficult and interesting processes that are being done by human being for many years. It can be taken as an art like any other fields of human creativity. A translation is said to be successful only if the output text has the same meaning as of the input text. Therefore, the transfer of lexical items and syntactic structures only is not considered successful translation if the overall meaning is not conveyed [10]. After computer scientists and linguists met with many failures in the beginning of MT application, they now understand the complexity of the task. Many researchers today are directing their efforts towards MT fully aware of the indistinctness of the vast task. MT has become a testing ground for many ideas in Computer Science, Artificial Intelligence and linguistics.

There are various challenges in machine translation; some of the major challenges that may obstruct and reduce the quality of the machine translation are described here.

1.4.1 Ambiguity

Ambiguity is one of the major sources of error in statistical machine translation. There are different types of ambiguities that may arise during the process of machine translation.

The *lexical ambiguity* may arise because of one of the following reasons or both [2]:

i) One word belongs to two or more lexical categories
ii) One word has more than one interpretation

Same word may be in different categories according to its context. For example, the word '*Fly'* can be used as noun in some context to denote an insect and it can be taken as verb in some other

context like in the sentence *'I like to fly'*. Similarly, in English same word can be interpreted differently in different sentences. For example, in the sentence;

I saw the film.

The meaning of the word '*film*' is either a movie or any thin layer like lather of the soap.

A *syntactic or structural ambiguity* is because of the two or more structural interpretation of the sentence. This type of ambiguity arises when we try to attach different prepositional phrases to different syntactical structure.

Sometimes sentence may give two or more meaning without the presence of lexical ambiguity or structural ambiguity such type of ambiguity is called *semantic ambiguity*. For example, in the sentence

This is metal case.

It is ambiguous that either it is a 'case' to store metal or it is the 'case' which is made up of metal.

1.4.2 Unknown Words

Unknown words are one of the great difficulties in the field of machine translation. According to [3], one of the most important problems of data-driven machine translation is that posed by unknown words: In the process of translation, a system is most likely to encounter words that were not present in the available training data. The Accuracy of the SMT degrades if the size of the training corpus is small. While this is in part due to the segmentation issues, it is also often simply due to the lack of training data. This problem increases exponentially if the size of the training corpus is small. Different techniques are developed to solve this problem of unknown words: like a heuristic based identification and translation method, a model based on morphological analysis and large amounts of lexical information contained in a dictionary, to estimate Part-Of-Speech information of unknown words using a statistical model of morphology and context etc. This thesis work is focused on this problem.

1.5 Proportional Analogies

Proportional analogies[16] are statements of the form;

$$A : B :: C : D$$

(read as "*A* is to *B* as *C* is to *D*") that shows the relationship between four entities . This statement can be taken as the equation which may give zero or more solution where one of the symbols used here is taken as a variable. Proportional analogies are one of the means to generate the knowledge of the worlds about some unknown things on the basis of some known things. Hence this representation is being popular in cognitive science as well as in artificial intelligence as in the examples below respectively [19]:

a) *electrons are to atoms as planets are to solar system*

b) *cat* : *kitten* :: *dog* : *puppy*

c) *speak* : *spoken* :: *break* : *broken*

In Linguistics, they are often used to explain historical language change, especially when previously unattested forms start to appear as in example "c" given above.

Chapter 2

PROBLEM DEFINITION

2.1 Background

Many changes and improvements are seen in recent years in the field of machine translation. A huge amount of researches are focused in statistical machine translation and example based machine translation. The area and scope of the machine translation is very large ranging from some domain specific system to domain less translation system. Although it is found that the statistical machine translation is more successful for some domain specific system [14], researchers are trying to make it useful for domain less system too. In different research projects in Europe and in the United States solutions for automatically translating Parliamentary speeches and broadcast news have been developed. In these contexts it is mandatory that the translation system should be effective to cover a variety of topic. More recently, the French-German project *Quaero* investigates possibilities to make use of machine translations for a multi-lingual internet. The project seeks to translate not only WebPages, but also videos and audio files found on the internet. The application of the machine translation for the country like Nepal is still more where the number literate people who can understand any foreign language are very less. Hence it will be more fruitful for those people if the various websites and technical manuals are available in Nepali language to increase their tendency of gaining knowledge from those websites as well as to increase their computer skills.

In spite of all these researches it is still a challenging job to solve the unknown word problem in SMT. It presents a great hurdle for Machine Translation. For low-resource languages, limited training data increases the frequency of unknown words and this degrades the quality of the translations. Presence of unknown words in the input sentence prevents the system from finding longer phrasal matches and produces low quality translations due to less reliable language model estimates [6]. In statistical machine translation to translate a sentence from one language to another a parallel corpus is used but it is not possible to a corpus to contain all the words from a whole language domain. It is estimated that the English language has a vocabulary of about 500,000 to 600,000 words with about 25,000 new words introduced each year [12]. In this scenario it is unlikely to any corpus to update all these words; hence the unknown word problem

is obvious. In most of the cases the unknown words are aligned to either "null" or discarded or left unchanged. For example,

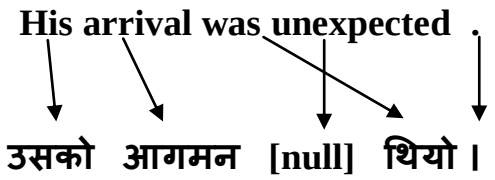

Fig 2.1: English to Nepali word level alignment including unknown word

In this example, the word '*unexpected*' is not contained in the source language (English) corpus hence this word remains un-translated. This type of problem increases if the size of the corpus is small. As mentioned earlier in the previous chapter, the translation process is not considered successful if the overall meaning is not preserved, which is restricted by these unknown words. Different approaches are used to solve this problem in machine translation and many of them are successful too. But very little work has been done in the context of English to Nepali machine translation.

Analogical reasoning is being the mean of gaining the knowledge in natural intelligence from ancient time. It involves several sub-processes: (1) retrieval of one case given another; (2) mapping between two cases in working memory; (3) evaluating the analogy and its inferences; and, sometimes, (4) abstracting the common structure. The core process in analogical reasoning is mapping. According to structure-mapping theory [7], an analogy is a mapping of knowledge from one domain (the base or source) into another (the target) such that a system of relations that holds among the base objects also holds among the target objects. In interpreting an analogy, people seek to put the objects of the base in one-to-one correspondence with the objects of the target so as to obtain the maximal structural match. The corresponding objects in the base and target need not resemble each other; what is important is that they hold like roles in the matching relational structures. Thus, analogy provides a way to focus on relational commonalities independently of the objects in which those relations are embedded.

The same thing can be true in the case of machines too. Given some cases the machine can be made capable of inferring some conclusion from those cases. This is one of the important aspects of the artificial intelligence (AI).

2.2 Literature Review and Related Works

Because of the haphazard nature of the previous work on the linguistic analogy, i.e., it is generally given a broader and more psychological definition, [16] gives a path that is more systematic way of dealing with words level analogy. It gives an algorithm to solve analogies in words. In analogies:

- Two different domains appear
- For both domains, modeling of a knowledge base in necessary
- Mapping of objects and transfer of properties are different operations
- The quality of analogies has to be evaluated as a function of the strength of properties transferred.

All of these should be simplified to answer the AI community by analogies.

To solve the word analogy *A : B = C : D* the algorithm verifies following constraint:

$$|A| = pdist\ (A, B) + pdist\ (A, C) + com\ (A\ , B, C, D)$$

Where, *pdist(a, b)* is the pseudo distance between two words *a* and *b,* and *com(a, b, c, d)* is the sum of the length of the common subsequences of words *a, b, c, d.*

Initially the value of *com (A, B, C, D)* is initialized to: $|A|$ - *(pdist(A,B) + pdist(A,C)).* i_A, i_B *and* i_C are the current positions in A, B and C. dir_{AB} and dir_{AC} are the direction of the path in matrix *AXB* and *AXC* from the current position respectively, then the algorithm to solve analogy is [16]:

if constraint(i_A, i_B, i_C, com(A, B, C, D))

case: dir_{AB} = dir_{AC} = diagonal

if A[i_A] = B[i_B] = C[i_C]

decrement com(A, B, C, D)

end if

copy B[i_B] + C[i_C] – A[i_A]

case: dir_{AB} = dir_{AC} = horizontal

copy char / min(pdist(A[1…i_A], B[1…i_B]), pdist(A[1…i_A], C[1…i_C]

case: dir_{AB} = dir_{AC} = vertical

move only in A (change horizontal line)

case: dir_{AB} != dir_{AC}

if dir_{AB} = horizontal

copy B[i_B]

else if dir_{AB} = vertical

move in A and C

else

same thing by exchanging B and C

end if

end if

Four different techniques for online handling of unknown words in phrase based statistical machine translation are described in [18], The techniques use morphological expansion (MORPHEX), spelling expansion (SPELLEX), dictionary word expansion (DICTEX) and proper name transliteration (TRANSEX) to reuse or extend phrase tables online.

In MORPHEX the out of vocabulary (OOV) word is matched with an in-vocabulary (INV) word that is a possible morphological variant of the OOV word. For this to work, it should be possible to analyze OOV word morphologically otherwise the technique will be failed. The morphological matching assumes the words to be matched agree in their lexeme but have different inflectional features. The information on possible inflectional variations from original phrase table is collected and analyzed single word source language is clustered that (a) translate into the same target language phrase and (b) have the same lexeme analysis. From these clusters it is learned that which morphological inflectional features in source language are irrelevant to target language. A rule set of morphological inflection maps is created that is used to relate analyses of OOV words to analyses of INV words (which is created off-line for speedy use).

In SPELLEX the OOV token is matched with an INV token that is a possible correct spelling of the OOV token. In [18], four types of spelling correction involving one letter only: letter deletion, letter insertion, letter inversion (of any two adjacent letters) and letter substitution is considered and letter substitution from a limited list of around 90 possible substitutions (as opposed to all 1260 possible substitutions) only is allowed. The substitutions considered include cases deemed harder than usual to notice as spelling errors: common letter shape alternations, phonological alternations and dialectal variations. The misspellings involving two words attached to each other or multiple types of single letter errors in the same word are not handled.

In DICTEX the phrase table with entries from a manually created dictionary is extended (the English glosses of the Buckwalter Arabic morphological analyzer in his case). For each analysis of an OOV word, the English lemma gloss is expanded to all its possible surface forms. The newly generated pairs are equally assigned very low translation probabilities that do not interfere with the rest of the phrase table.

TRANSEX is the technique that is used to translate the proper name. The target language transliteration hypothesis is produced that assumes the OOV is a proper name. The transliteration system is rather simple: it uses the transliteration similarity measure described by [5] to select a best match from a large list of possible names in English. For each OOV word, a list of possible transliterations that are used to add translation pair entries in the phrase table is produced. The

newly generated pairs are assigned very low translation probabilities that do not interfere with the rest of the phrase table. Weights of entries were modulated by the degree of similarity indicated by the metric we used. Given the large number of possible matches, only the top 20 matches are passed to the phrase table.

Example Based Machine Translation (EBMT) using Proportional Analogy, a type of analogical learning, was attractive because of its simplicity; and the papers reported considerable success with the method. Authors of [19] review what is believed to be the totality of research reported using this method, as an introduction to their own experiments in this framework, reported in a companion paper. They report first some lack of clarity in the previously published work, and then report their findings that the purity of the proportional analogy approach imposes huge run-time complexity for the EBMT task even when heuristics as hinted at in the original literature are applied to reduce the amount of computation.

Using the algorithm from [16] for the purpose of EBMT, a database of example *pairs* is assumed, where each sentence has a corresponding translation. For example; in the analogy

They went to school : They came home = He went to school : He came home

Suppose the translation of first three sentences is:

a) तिनीहरु विद्दयालय गए

b) तिनीहरु घर आए

c) ऊ विद्दयालय गयो

If the sentence *he came home* is to be translated then the translation process is as follow

i) Find a triplet of sentences in the example set that satisfy the analogical equation
A : B = C : *he came home*

ii) Take the translations corresponding to *A*, *B* and *C* (notated *A*', *B*', *C*').

iii) Solve the analogical equation
A' : B' = C' : ***x,*** ***x*** represents the desired translation.

Substituting the three sentences in above equation, we have a solvable analogical equation with ***x*** = *ऊ घर आयो*, which is an acceptable translation.

But [19] points out two potential problems in this process:

1) First is that for a given input sentence *D*, the database may contain multiple triples (*A*, *B*, *C*) that offer a solvable analogy.
2) The second problem, also due to the unconstrained nature of the mechanism, multiple solutions may be produced.

To reduce these problems, [19] purposes some heuristics to be used, some of them are:

1. Consider as candidates only sentences whose length is more than half and less than double the length of the input sentence.
2. Consider as candidates primarily sentence pairs where *A* has a low edit distance with respect to *D*, and *B* has a low inclusion score.
3. Consider as candidates primarily sentence pairs where *A* and *B* have the largest LCS (or significant *n*-grams) in common with *D*, starting with *A*s and *B*s that share the longest LCS (or significant *n*-grams) with each other, and with LCSs of a similar length.
4. Consider as candidates only pairs where *B* or *C* share the same first or last symbol with *A* or *D*.
5. Whenever a symbol occurs more frequently in *A*' than it does in *B*' and *C*', the equation is bound to fail and need not be solved.

A unified definition for the notion of (formal) analogical proportion, which applies to a wide range of algebraic structures, is proposed by [17]. The authors show that this definition is suitable for learning in domains involving large databases of structured data, as is especially the case in Natural Language Processing (NLP). The authors then present experimental results obtained on two morphological analysis tasks which demonstrate the flexibility and accuracy of their approach. It demonstrates the flexibility of the analogical learner. Two different supervised learning tasks are considered, both aimed at performing the lexical analysis of isolated word forms. Each of these tasks represents a possible instantiation of the learning procedure.

The first experiment in [17] consists in computing one or several vector(s) of morphosyntactic features to be associated with a form. Each vector comprises the lemma, the part-of-speech, and based on the part-of-speech, additional features such as number, gender, case, tense, mood, etc. The second experiment consists in computing a morphological parse of unknown lemmas: for

each input lemma, the output of the system is one or several parse trees representing a possible hierarchical decomposition of the input into (morphologically categorized) morphemes. This kind of analysis makes it possible to reconstruct the series of morphological operations deriving a lemma, to compute its root, its part-of-speech, and to identify morpheme boundaries. This information is required, for instance, to compute the pronunciation of an unknown word; or to infer the compositional meaning of a complex (derived or compound) lemma. Bins gather entries sharing a common root.

In [3] a method is proposed to use proportional analogy at the character-level to translate unknown words. The study and report result of the integration of this approach into a statistical machine translation system translating from Japanese to English with relatively scarce resources. Objective evaluation measures suggest that the translated sentences have a higher adequacy than that produced by a baseline system, while their fluency is similar.

In the context of statistical machine translation a bilingual corpus of training data is used to produce word alignments. Those word alignments may then be used in the form of a lexical translation table, in order to extract consistent phrase pairs. However, if at translation time a word or word sequence is not found in the phrase table, no translation may be retrieved. The tokens remain untranslated, which considerably degrades translation quality.

A method in order to alleviate unknown word problem is proposed in [3] to apply. Suppose that a bilingual corpus as training data is available for a statistical machine translation system, and a test set consisting of sentences in the source language, that it is wished to be translated in the target language. Given the available training data, a statistical machine translation system is built, and the test set is translated. The machine translation output includes unknown words that are automatically extracted and gathered as the *unknown words set*. Now this unknown word set is translated using proportional analogy with the help of bilingual data. While the given training corpus may be used as is to directly translate, we wish to use additional aligned data that is consistent in terms of size with what we wish to translate: tokens consisting in short character strings. This additional data may be extracted from the lexical translation table that was estimated from statistical word alignments when building the statistical machine translation

system. While low-probability alignments may be discarded, extracting the *N*-first alignments for each source target word allows to conveniently building a basic dictionary.
It is shown in [13] that analogical learning offers as well an elegant and effective solution to the problem of identifying potential translations of unknown words.

A learning set $L = \{L_1, \ldots, L_N\}$ gathers N observations. A set of features computed on an incomplete observation *X* defines an input space. The inference task consists in predicting the missing features which belong to an output space. *I(X)* and *O(X)* are used to denote the projection of *X* into the input and output space respectively. The inference procedure involves three steps [13]:

1. Building $E_I(X) = \{(A,B,C) \in L^3 \mid [I(A) : I(B) = I(C) : I(X)]\}$, the set of input stems of *X*, that is the set of triplets (A,B,C) which form with *X* an analogical equation.
2. Building $E_O(X) = \{Y \mid [O(A): O(B) = O(C) : Y\ ], (A,B,C) \in E_I(X)\}$, the set of solutions to the analogical equations obtained by projecting the stems of $E_I(X)$ into the output space.
3. Selecting *O(X)* among the elements of $E_O(X)$.

This implementation is also based on [15] algorithm for computing the solutions of a formal analogical equation [A : B = C : ?]. This requires computing two edit-distance tables, one between A and B and one between A and C. It seeks for the subsequences of B and C which is not common subsequence of A, for this in the edit-distance table the insertion cost is null. Then the algorithm matches the alignments according to the paths of minimum cost in each table between A and B, & between A and C as shown in the figure 2.2(a) [13], where A = *even*, B = *usual* and C = *unevenly* In this example, there are 681 different paths that align even and usual (with a cost of 4), and 1 path which aligns even with unevenly (with a cost of 0). This result in 681 synchronizations which generates 15 different solutions, among which only *unusually* is a legitimate word-form (shown in figure 2.2(b)) [13].

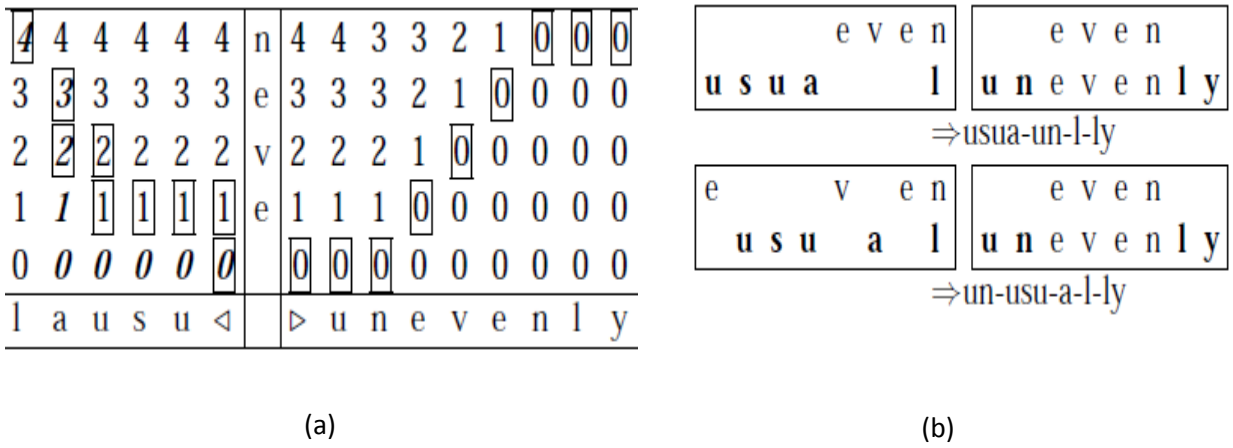

Fig 2.2 : Edit-distance tables computed between 'even' and 'usual' & 'even' and 'unevenly' along with synchronizations computed while solving the equation [even : usual = unevenly : ?].

A new algorithm to solve the unknown word problem in machine translation using the online dictionary definition of those words is proposed by [4]. In this method instead of translating the unknown word itself, the meaning of those unknown words in source language are identified from any online dictionary and translated into the source language. Only monolingual resources are required, which generally offer a broader coverage than bilingual resources and are available for a large number of languages.

In this process given an unknown word first of all the unambiguous and useful definition is identified in the source language such that the definition itself doesn't contain any unknown word. The extracted definition is then translated into the target language. The major problem here is the possible occurrence of unknown word in the definition. In such cases the unknown words are again subjected to the system to extract the definition in recursive fashion. Another problem is that the style of the definitions can be very different from the domain of the actual translation system like the definition of any unknown word may not be necessarily a single word. In such cases the translation system is trained on dialogs, a completely different style than the short and concise definitions.

Finally the translated definition is inserted into the baseline translation. It then clearly marks the definition as such and leaves the decision if that definition clearly defines a word or a short phrase to the speaker of the target language to avoid affecting the coherence of the rest of the sentence. The similar approach of using online dictionary to handle the unknown words in English to Nepali statistical machine translation is used by [11] recently.

Different strategies for identifying analogies in a input space is discussed by [15]. It proposes a data-structure and algorithms that allow controlling the balance between speed and recall. For very high-dimensional input spaces (hundreds of thousands of elements), it proposes a heuristic which reduces computation time with a limited impact on recall.

Chapter 3
IMPLEMENTATION

This approach for unknown word handling is based on the algorithm proposed by Yves Lepage in [16] to solve analogical equation. It is similar to the approaches described in [13] and [3] but some heuristics used here make it different from the previous works.

In the analogical reasoning approach an equation of the form A : B = C : D is identified where A, B and C are some known words and D is any unknown word. Solving this equation may give the multiple solutions for D. Some heuristics to limit the number of possible solutions of the equation has been used. As mentioned earlier that the English language has a vocabulary of about 500,000 to 600,000 words and 25,000 of new words are introduced each year. Among these new words some of the words are derived from some other languages as they are, like the word '*avata*r' is derived from Sanskrit, meaning '*incarnation*', similarly a large volume of new words are formed by adding some prefixes and suffixes on existing words. It is not possible to any corpus to contain all of these words from that language domain.

Hence, a method for solving the unknown words on the basis of these prefixes and suffixes for English to Nepali machine translation is proposed here. Prefixes are added to some root words (stems) at the beginning and suffixes are added at the end of the word. These prefixes and suffixes give some other meaning to those words and the meaning of those prefixes and suffixes are often remains same for the all words in case of Nepali language too. This property of prefixes and suffixes is used here to translate unknown word. It is considered that the original stem of the unknown word having some prefix or suffix is already in our corpus along with some other words with same prefix or suffix with its stem in that corpus. With the help of meaning of these three words the meaning of the unknown word is identified.

3.1 Phases of Implementation

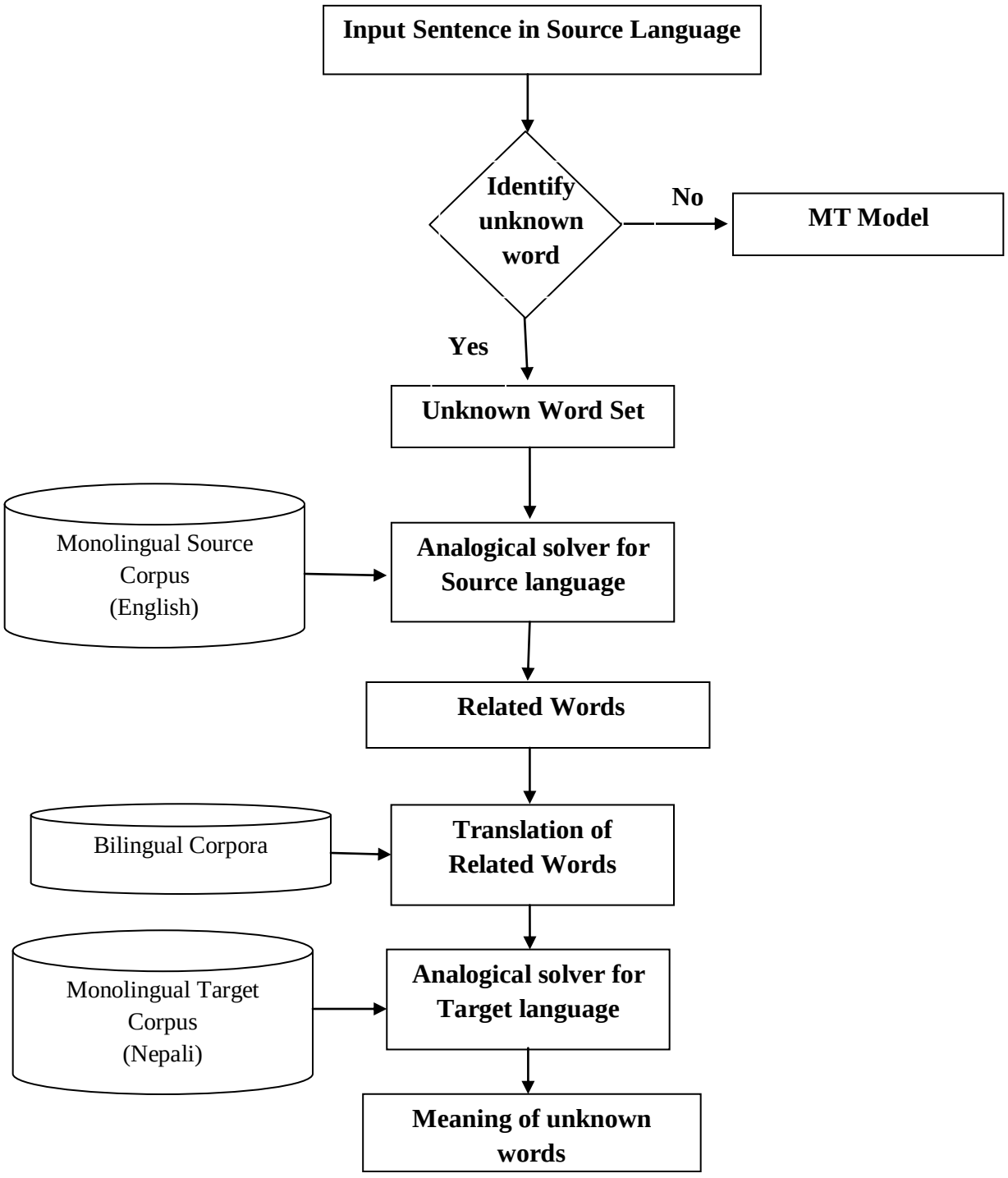

Fig 3.1: Implementation model for handling unknown word

3.1.1 Identify unknown words

In this phase the given input sentence is tokenized in words and the English parallel corpus is searched for all of these words. If all of these words are contained by the corpus, there is nothing to be done since there is not any unknown word and hence passed to the machine translation system but if any of these words is not in the corpus then it is taken as unknown word and hence passed to the analogical solver to identify its related words in corpus.

3.1.2 Solving analogical equation in source language

After identifying the unknown word it is subjected to the analogical solver to identify its related word (i.e. the stem). For this purpose the longest common subsequence algorithm is used. The system searches the whole corpus for the word having longest common subsequence with the unknown word. For example if "*immortal*" is the unknown word that we tried to find the meaning for, then the system searches for its stem and identifies the word "*mortal*" as its related word as shown in the figure 3.2:

	m	***o***	***r***	***t***	***a***	***l***
i	0	0	0	0	0	0
m	1	1	1	1	1	1
m	1	1	1	1	1	1
o	1	2	2	2	2	2
r	1	2	3	3	3	3
t	1	2	3	4	4	4
a	1	2	3	4	5	5
l	1	2	3	4	5	6

Fig 3.2: calculation of longest common subsequence between words 'mortal' and 'immortal'.

Here, the length of longest common subsequence between *immortal* and *mortal* is 6. If the length of the longest common subsequence is less than 3 then the word is discarded and hence it is supposed that not any related word is found and the system fails, in this case the system gives no output. Otherwise the word having longest common subsequence with unknown word is taken as

the related word. Now, to identify the remaining two words the system searches for the word having same suffix or prefix as of the unknown word and its stem too. And the meanings of these three words are identified using SMT system.

3.1.3 Translating the word triplets

The word triplets identified in the previous phase are now translated to target language words using SMT model which uses the Expectation Maximization (EM) algorithm to identify the corresponding meaning of those words. As the SMT model the system designed and described in [2] is used here. As for the above example if the related word triplet is

possible : impossible = mortal : x

Then it is translated in Nepali language as

सम्भव : असम्भव = मरणशिल : x

This is in the form of the formal analogical equation that we have to solve to find the meaning of unknown word.

3.1.4 Solving analogical equation in target language

After translating the analogical equation from source language to target language of the form A : B = C : x, it is searched for the substrings which are not common is A & B and A & C and those substrings are arranged in some acceptable form. For the example given above

सम्भव : असम्भव = मरणशिल : x

The substring which is not common in A and B is "अ" and the substring that is not common in A and C is "मरणशिल". After combining these substrings it forms either "अमरणशिल" or "मरणशिलअ". But as the matter of heuristic, a technique is used that if in the first word pair the not common substring is prefix, it is added as prefix in third word and if in the first word pair the not common substring is suffix it is added as suffix. Hence the solution for this analogical equation is

सम्भव : असम्भव = मरणशिल : x => अमरणशिल

Chapter 4

TESTING AND ANALYSIS

4.1 Testing and Training Data

To train the system with EM algorithm various words with different prefixes and suffixes were selected randomly from Ajanta's English-Nepali dictionary and some other words having same prefixes or suffixes were searched with their root words. A parallel corpus was created using those words for training purpose and the originally selected words were used as unknown words for testing.

4.1.1 Training

English parallel corpus
human is mortal
that is impossible
we ate continental food in that restaurant
the intercontinental alliance attacked there
this is not possible
that was an terrorist act
these are all known words
his work was satisfactory
unknown word translation is possible
the pilot landed the plane safely
he is a tourism worker
I am with my coworker
they have several active nuclear reactors
we have to moralize him
they demoralize the children
he is my good friend
I am the citizen of this nation
our friendship is strong

as the counteract of terrorism they attacked other nation
this is natural act
my nationality is Nepali
This is a national monument
it is soft like butter
feel the softness
he is very kind person
we believe in customer satisfaction
you may disagree on this
I agree with you
this is a regular phenomena
he is a socialist leader
that was the origin of socialism
these are our historical monument
everything was planned
the picnic was not preplanned
They are husband and wife
I met an ex-minister
he is a minister
It is reliable
this was not known
my result is satisfactory
there was an unknown person
he was very active in his student life

Table 4.1: English parallel corpus used to train the system.

Nepali parallel corpus
मानिस मरणशिल छ
त्यो असम्भव छ

हामीले त्यो रेस्टुरेन्टमा महादेशिय खाना खायौं

अन्तरमहादेशिय गठबन्धनले त्यहाँ आक्रमण गर्यो

यो सम्भव छैन

त्यो एउटा आतकंवादी कृया थियो

यि सबै ज्ञात शब्दहरु हुन्

उसको काम सन्तोषजनक थियो

अज्ञात शब्द रुपान्तरण सम्भव छ

चालकले विमान सुरक्षित अवतरण गरायो

ऊ एउटा पर्यटन कर्मि हो

म मेरो सहकर्मि सँग छु

तिनिहरु सँग धेरै सकृय आणविक भट्टिहरु छन्

हामिले उसलाई उत्साहित गर्नु पर्छ

तिनिहरुले वच्चाहरुलाई निरुत्साहित गरे

ऊ मेरो राम्रो मित्र हो

म यो राष्ट्र को नागरिक हुँ

हाम्रो मित्रता बलियो छ

आतंकवाद को प्रतिकृया स्वरुप तिनिहरुले अरु राष्ट्रमा आक्रमण गरे

यो प्राकृतिक कृया हो

मेरो राष्ट्रियता नेपाली हो

यो राष्ट्रिय सम्पदा हो

यो नौनी जस्तै नरम छ

नरमपन अनुभव गर
ऊ धेरै दयालु व्यक्ति हो
हामी ग्राहकको सन्तुष्टि मा विश्वास गर्छौं
तिमी यसमा असहमत हुन सक्छौ
म तिमि सँग सहमत छु
यो नियमित प्रकृया हो
ऊ एउटा समाजवादी नेता हो
त्यो समाजवाद को उत्पत्ति थियो
यिनिहरु हाम्रा ऐतिहासिक सम्पदा हुन्
सबैकुरा नियोजित थियो
वनभोज पूर्वनियोजित थिएन
उनीहरु पति र पत्नी हुन्
मैले एक पूर्वमन्त्री लाई भेटें
उ एक मन्त्री हो
यो विश्वसनिय छ
यो ज्ञात थिएन
मेरो नतिजा सन्तोषजनक छ
त्यहाँ एउटा अज्ञात व्यक्ती थियो
ऊ आफ्नो विद्यार्थि जिवन मा धेरै सकृय थियो

Table 4.2: Nepali parallel corpus used to train the system

4.1.2 Testing

Input Sentence	Unknown Word	Expected Output	Output	Remarks
No one is immortal	Immortal	अमरणशिल	अमरणशिल	
I want to interact with him	Interact	अन्तरकृया	---------	Selection of wrong word
The result was unsatisfactory	Unsatisfactory	असन्तोषजनक	असन्तोषजनक	
He was copilot	Copilot	सहचालक	---------	Selection of wrong word
I have Nepali citizenship	Citizenship	नागरिकता	नागरिकता	
It is counterterrorist act	Counterterrorist	प्रतीआतकंवादी	---------	Selection of wrong word
you have to give regularity to this work	Regularity	नियमितता	नियमितता	
It is his kindness	kindness	दयालुपन	---------	Selection of wrong word
There is no dissatisfaction	Dissatisfaction	असन्तुष्टि	असन्तुष्टि	
It is prehistorical monument	Prehistorical	पूर्वऐतिहासिक	पूर्वऐतिहासिक	
She is his ex-wife	Ex-wife	पूर्वपत्नी	पूर्वपत्नी	
This is social phenomena	Social	सामाजिक	---------	No output
Deactivate the reactor	deactivate	निस्कृय	---------	No output
It shows our reliability	Reliability	विश्वसनियता	विश्वसनियता	

Table 4.3: Testing system with different unknown words

4.2 Analysis

The accuracy of the system do not depends entirely on unknown word handling model but it also depends on the accuracy of the SMT system too. The system was tested with 14 randomly selected unknown words; among them for 8 words, the expected output was found and remaining words remained untranslated because of some false analogy[4] and incorrect result of SMT.

Here,

The number of unknown words provided= 14

Number of correct output= 8

Therefore, accuracy = (8/14) * 100 % = 57.14 %

4.3 Verification and validation

Verification shows our system is giving correct result for all the instances of data from the expected input domain and validation is used to show the system meets all the requirements. This system was validated with multiple data which shows that it meets our requirement of translating unknown words which are formed by adding some prefixes or suffixes on the word stem which are already in the corpus. It was verified for 57.14 % of provided input data.

[4] Sometimes the system selects some unrelated word because the length of the longest common subsequence of that word is more than the actual stem of unknown word, resulting in false analogy.

Chapter 5
CONCLUSION AND FURTHER STUDY

5.1 Conclusion

Handling unknown word is one of the major challenges in statistical machine translation. However many approaches are used to translate the unknown words, the analogical learning is one of the most effective approaches [13]. It is the human nature that every human can learn new things everyday by studying its surrounding and infer some solution and conclusion about something that is not known to him or her previously. The same approach is found effective in the case of AI and machine translation too, to handle the unknown word. Translating unknown words using this approach gives the considerable amount of accuracy to machine translation. The accuracy of the translation is increased if the system is specific to some domain because if a corpus contains the word from a specific domain like medical reports, travel guide etc. then there is more chances that the corpus contains the word root for any unknown word.

Since, more words in the training corpus reduce the number of unknown words; the accuracy of the translation can be increased by increasing the size of training corpus. But along with the number of words the amount of false analogy may increase hence the appropriate size of training corpus is required.

5.2 Further Study

One of the major problems in this approach of handling unknown words is due to the "false analogy". While selecting the related words for any unknown word using longest common subsequence technique, sometimes the system may select some other word having longest common subsequence with it instead of the actual stem of the word. For example, in our system, for the unknown word "interact" the system selects the word "intercontinental" as the related word instead of its root word "act". This leads the system to some false analogy and degrades the quality of translation. To solve this, it is necessary to maintain the corpus such that there will remain as much as less words in the training corpus which may relate to a given unknown word.

But this will again increase the chances of occurring unknown word problem hence it is necessary to identify the optimal size of the training corpus.

Although this approach has shown the considerable amount of accuracy, It is unable to translate the name entities hence an appropriate technique to handle named entity should be used.

Appendices

Appendix A

Code to tokenize the sentence:

```
String in =intxtfld.getText();
String token[]=in.split(" ");
String unknown;
String relWord="";
String relWord1="";
String relWord2="";
int flag = 0;
try
{
   for(int i=0; i<token.length; i++)
   {
  Scanner input = new Scanner(new File("d:/paralalEng.txt"));
  while(input.hasNext())
   {
      String test = input.next();
      if(token[i].equalsIgnoreCase(test))
      {

         flag = 1;
         break;
      }
   }
  if(flag == 0)
   {
      unknown=token[i];
      System.out.println(unknown);
      String arg[] = new String[2];
      arg[0] = unknown;
```

```
        relWord=PairWords.main(arg);
        arg[1]=relWord;
        relWord1=LCS.main(arg);
        arg[0]=relWord1;
        relWord2=PairWords.main(arg);
    }
   flag = 0;
    }
    PrintStream out = new PrintStream(new File("d:\\test.txt"));
    out.print(relWord + " " + relWord1 + " " + relWord2 );
    out.close();
    FindCoressNepaliWords.main(null);
    FindTheMeaningOfUnknownWord.main(null);
}
 catch(Exception e)
 {
    System.out.println(e);
 }
```

Appendix B

Code to identify the root word:

```
int maxLen=0;
 String relWord="";
 Scanner fin = new Scanner(new File("d:/paralalEng.txt"));
 while(fin.hasNext())
 {
    String s = fin.next();
    if(!s.equalsIgnoreCase(arg[0]))
    {
    int len = LCS.getLCS(arg[0], s);
    if(len>maxLen)
    {
       maxLen=len;
       relWord=s;
    }
    }
 }
fin.close();
 return relWord;
```

Appendix C

Code for longest common subsequence:

```
int storeLength [] = new int[100];
String storeString [] = new String[100];
public static ArrayList<String> list = new ArrayList<String>();
public static ArrayList<Integer> listLength = new ArrayList<Integer>();
public static char printLCS(String[][] b, String str1, int i, int j, String ans)
  {
    if(i == 0 || j == 0)
    {
      if(ans.length() >=3)
      {
        list.add(ans+":"+ans.length());
        listLength.add(ans.length());
      }
      return '0';
    }
    if(b[i][j].equalsIgnoreCase("ul"))
    {
      ans = ans + String.valueOf(str1.charAt(i-1));
      printLCS(b, str1, i-1, j-1, ans);
      }
    else if(b[i][j].equalsIgnoreCase("u"))
    {
      printLCS(b, str1, i-1, j, ans);
    }
    else
    {
      printLCS(b, str1, i, j-1, ans);
    }
    return '0';
```

```
    }
    public static int getLCS(String str1, String str2)
    {
        int m = str1.length();
        int n = str2.length();
        String b[][] = new String[m+1][n+1];
        int c[][] = new int[m+1][n+1];
        for(int i=0; i<m; i++)
         {
            for(int j=0; j<n; j++)
            {
               if(str1.charAt(i) == str2.charAt(j))
               {
                  c[i+1][j+1] = c[i][j] + 1;
                  b[i+1][j+1] = "ul";
               }
               else if(c[i][j+1] >= c[i+1][j])
               {
                 c[i+1][j+1] = c[i][j+1];
                 b[i+1][j+1] = "u";
               }
               else
               {
                  c[i+1][j+1] = c[i+1][j];
                  b[i+1][j+1] = "l";
               }
            }
         }
        printLCS(b, str1, m, n, "");
        return c[m][n];
}
```

Appendix D

Code to find the related words:

```
public static String main(String arg[]) throws Exception
  {
     String relWord=arg[1];
     String relWord1="";
     String searchStrPrefix = "";
     String searchStrSuffix = "";
     int flag1 = 0;
     int flag2 = 0;
     if(arg[0].endsWith(relWord))// unknown word fromed by adding some prefix to known word
     {
        int len = relWord.length();
        System.out.println(arg[0].substring(0, arg[0].length()-len));
        searchStrPrefix = arg[0].substring(0, arg[0].length() - len);
        flag1 = 1;
        System.out.println("flag1 = " + flag1);
     }
   if(arg[0].startsWith(relWord))// unknown word formed by adding some suffix to known word
     {
        int len=relWord.length();
        System.out.println(arg[0].substring(len, arg[0].length()));
        searchStrSuffix = arg[0].substring(len, arg[0].length());
        flag2 = 1;
        System.out.println("flag2 = " +flag2);
     }
     Scanner filin = new Scanner(new File("d:/paralalEng.txt"));
     if(flag1 == 1 && flag2 ==  1)
     {
        while(filin.hasNext())
```

```
    {
        String s = filin.next();
        if(s.startsWith(searchStrPrefix) && s.endsWith(searchStrSuffix))// unknown word is formed by adding both prefix and suffix to known word
        {
            relWord1=s;
            break;
        }
    }
}
else if(flag1 == 1)
{
    while(filin.hasNext())
    {
        String s = filin.next();
        if(s.startsWith(searchStrPrefix))
        {
            relWord1=s;
            break;
        }
    }
}
else if(flag2 == 1)
{
    while(filin.hasNext())
    {
        String s = filin.next();
        if(s.endsWith(searchStrSuffix))
        {
            relWord1=s;
            break;
```

```
            }
        }
    }
    filin.close();
    System.out.println("relword1 = "+ relWord1);
    return relWord1;
}
```

Appendix E

Finding corresponding Nepali word with EM algorithm:

```
   public ArrayList<SentencePair> parallelSentence;
   public Hashtable<String,SourceTargetWordPairs> wordtable;
   public Hashtable<String,Double> translationProbability;
   public static Hashtable<String,String> tblForEnglishChunk = new Hashtable<String, String>();
   public static Hashtable<String,String> tblForNepaliChunk = new Hashtable<String, String>();
    //tbl to find the maximum probability
   public Hashtable<Double,String> tblToFindMostProbableChunk = new Hashtable<Double, String>();
   //Arraylist to put the answer
   ArrayList<String> listHoldingTheAlignedChunk = new ArrayList<String>();

   //table to hold the english chunk and coressponding nepali chunks
   public Hashtable<String, String> tblForMostProbableChunk = new Hashtable<String, String>();
/*************************************************************************/
   public static void main(String arg[]) throws Exception
   {
                FindCoressNepaliWords a = new FindCoressNepaliWords();
                a.makeParallelCorpus();
                a.makeWordPair();
                a.displayWordPair();
                a.EMalgorithm();
}

/*************************************************************************/
```

```
public void makeParallelCorpus() throws IOException
	{
		Corpus c = new Corpus();
		String englishSentences[];
		String nepaliSentences[];
		englishSentences = c.getEnglishcorpus();
		nepaliSentences = c.getNepaliCorpus();
		parallelSentence = new ArrayList<SentencePair>();
		for(int i=0;i<englishSentences.length;i++)
		{
			SentencePair snt;
			String sTokens[] = tokenPopulate(englishSentences[i]);
			String tTokens[] = tokenPopulate(nepaliSentences[i]);
			snt = new SentencePair(sTokens,tTokens);
			this.parallelSentence.add(snt);
		}
	}
  /*****************************************************************************/
	public String[] tokenPopulate(String sentence)
	{
		int i = 0;
		String tokens[];
		tokens = sentence.split(" ");
		return tokens;
	}
 /*****************************************************************************/
	public void makeWordPair()//collect the word types from source and related words
                                                        //from target corpora
	{
		wordtable = new Hashtable<String,SourceTargetWordPairs>();
		for(int i=0;i<parallelSentence.size();i++)
```

```
                {
                        SentencePair snt = parallelSentence.get(i);
                        String[] sourceSentence = snt.getSourceSentence();
                        for(int j=0; j<sourceSentence.length; j++)
                        {
                        HashSet<String> targetWords = findTargetWords(sourceSentence[j]);
                        SourceTargetWordPairs stwpairs = new
SourceTargetWordPairs(sourceSentence[j],targetWords);
            this.wordtable.put(sourceSentence[j],stwpairs);
                        }
                }
        }
  /*************************************************************************************/
        public HashSet<String> findTargetWords(String sword)
        {
                HashSet<String> targetWords = new HashSet<String>();
                for(int i=0;i<parallelSentence.size();i++)
                {
                        SentencePair snt = parallelSentence.get(i);
                        String[] sourceSentence = snt.getSourceSentence();
                        String[] targetSentence = snt.getTargetSentence();
                        for(int j=0; j<sourceSentence.length; j++)
                        {
                                if(sourceSentence[j].equalsIgnoreCase(sword))
                                {
                                        for(int k=0; k<targetSentence.length;k++)
                targetWords.add(targetSentence[k]);
                                }
                        }
                }
                return targetWords;}
```

```
/*************************************************************************/
        public void displayWordPair()
        {

                String[] strs = wordtable.keySet().toArray( new String[0] );
                for(int i=0; i<this.wordtable.size(); i++)
                {
                        SourceTargetWordPairs wp = this.wordtable.get(strs[i]);
                        System.out.print(wp.getSourceWord()+":\t");
                        System.out.print(wp.getTargetWords().toString()+"\n");
                }
        }

  /*************************************************************************/
public void EMalgorithm()
        {
                String sword;
                String tword;
                Hashtable<String,Double> translationCount = new Hashtable<String,Double>();

                translationProbability = new Hashtable<String,Double>();
                SourceTargetWordPairs[] st = wordtable.values().toArray(new
SourceTargetWordPairs[0]);
                for(int i=0; i<st.length; i++)
                {
                        sword = st[i].getSourceWord();
                        Object[] targetWords = st[i].getTargetWords().toArray(new Object[0]);
                        //Calculate the word pair probability
                        double tProb = (double) 1/st[i].getTargetWords().size();
                        for(int j=0; j<targetWords.length; j++)
                        {
```

```
                    tword = targetWords[j].toString();
                    WordPair wp = new WordPair(sword,tword);
                    //Initialize the translation probability to the HashTable
                    translationProbability.put(wp.toString(),tProb);
                }
            }
            /*********************************************/
            for(int n=0;n<3;n++)
            {
WordPair[] WordPairObjects = new WordPair[translationProbability.size()];
for(int i=0; i<translationProbability.size(); i++)
   WordPairObjects[i] = new WordPair();
                for(int i=0;i<WordPairObjects.length;i++)
                {
                    translationCount.put(WordPairObjects[i].toString(),0.0);
                }//end for

                for(int s=0;s<parallelSentence.size();s++)//for each sentence pair
                {
                    SentencePair snt =parallelSentence.get(s);
                    String[] targetSentence = snt.getTargetSentence();
                    String[] sourceSentence = snt.getSourceSentence();
                    for(int j=0;j<targetSentence.length;j++)
                    {
                        double total = 0.0;
                        tword = targetSentence[j];

                        for(int i=0;i<sourceSentence.length;i++)
                        {
                            double d = 0;
                            sword = sourceSentence[i];
```

```
                                        WordPair wp = new WordPair(sword,tword);
                                        if(translationProbability.get(wp.toString()) == null)
                                                d = 0;
                                        else
                                        d = translationProbability.get(wp.toString());
                                         total += d;
                                        for(int k=0;k<sourceSentence.length;k++)
                                        {
                                                double tpValue = 0;
                                                 double tcValue = 0;
                                                String sourceWord = sourceSentence[k];
                                                WordPair wpObject = new ceWord,tword);
                                    if(translationProbability.get(wpObject.toString()) != null)
                        tpValue = translationProbability.get(wpObject.toString());
                       if(translationCount.get(wpObject.toString()) != null)
                          tcValue = translationCount.get(wpObject.toString());
                    tcValue += (tpValue/total);
                    translationCount.put(wpObject.toString(),tcValue);
                                    }//end for
                            }//end for
                    }//end for
            }//end for
            //re-estimate the translation parameter(translationProbability) values
            for(int s=0;s<parallelSentence.size();s++)//for each sentence pair

{
                                    SentencePair snt =parallelSentence.get(s);
                                    String[] sourceSentence = snt.getSourceSentence();
                                    String[] targetSentence = snt.getTargetSentence();
                                    for(int i=0;i<sourceSentence.length;i++)
                                    {
```

```
                                        double total = 0;
                                        sword = sourceSentence[i];

                                        for(int j=0;j<targetSentence.length;j++)
                                        {
                                                tword = targetSentence[j];
                                                WordPair wp = new WordPair(sword,tword);
                                                total += translationCount.get(wp.toString());
                                                for(int k=0;k<targetSentence.length;k++)
                                                {
                                                        double tcValue = 0;
                                                        double tpValue = 0;
                                                        String targetWord = targetSentence[k];
                                                WordPair wpObject = new WordPair(sword,targetWord);
                                                  if(translationCount.get(wpObject.toString()) != null)
                                                 tcValue = translationCount.get(wpObject.toString());
                                            if(translationProbability.get(wpObject.toString()) != null)
                                                 tpValue = translationProbability.get(wpObject.toString());
                                        tpValue = tcValue/total;
                                        translationProbability.put(wpObject.toString(),tpValue);
                                                }//end for
                                        }//end for
                                }//end for
                        }

        }
//EM algorithm ends here
```

Appendix F

Finding the meaning of unknown word:

```
public static void main(String arg[]) throws Exception
  {
     Charset ch = Charset.forName("UTF-8");
     FileInputStream ncorpus =  new FileInputStream("d:\\nepWords.txt");
     InputStreamReader irn = new InputStreamReader(ncorpus,ch);
     BufferedReader nr = new BufferedReader(irn);
     String str = "";
     while(true)
     {
        str = nr.readLine();
        break;
     }
     ncorpus.close();
    String s[] = str.split(" ");
     int len1 = s[2].length();
     int len2 = s[3].length();
     String strPrefix = "";
     String strSuffix = "";
     String str1 = "";
     int flag1 = 0;
     int flag2 = 0;
     if(s[2].endsWith(s[3]))
     {
        strPrefix = s[2].substring(0, len1-len2);
        flag1=1;
     }
     if(s[2].startsWith(s[3]))
     {
        strSuffix = s[2].substring(len2, len1);
```

```
            flag2=1;
        }
            Charset ch1 = Charset.forName("UTF-8");
            FileOutputStream ncorpus1 =  new FileOutputStream("d:\\nepWords1.txt");
            OutputStreamWriter irn1 = new OutputStreamWriter(ncorpus1,ch1);
            BufferedWriter nr1 = new BufferedWriter(irn1);
 if(flag1 == 1 && flag2 == 2)
        {
            str1 = strPrefix.concat(s[1]);
            str1 = str1.concat(strSuffix);
        }
  if(flag1 == 1)
        {
          str1 = strPrefix.concat(s[1]);
        }
  if(flag2 == 1)
        {
            str1 = s[1].concat(strSuffix);
        }
   nr1.write(str1);
  nr1.close();
}
```

References

[1] Attia Mohammed A., *Handling Arabic Morphological and Syntactic Ambiguity within the LFG Framework with a View to Machine Translation,* PhD thesis, University of Manchester, 2008

[2] Balami Bikash, *A Chunk Level Statistical Machine Translation (English Language to Nepali Language Translation),* Master Thesis, Tribhuvan University., 2009

[3] Denoual Etienne, *Analogical translation of unknown words in a statistical machine translation framework,* 2007

[4] Eck Matthias, Vogel Stephan, Waibel Alex, *Communicating Unknown Words in Machine Translation,* 2008

[5] Freeman A., Condon S., and Ackerman C., *Cross Linguistic Name Matching in English and Arabic,* 2006

[6] Gangadharaiah Rashmi, Brown Ralf D., Carbonell Jaime, *Monolingual Distributional Profiles for Word Substitution in Machine Translation*, 2010

[7] Gentner, Dedre. "Are Scientific Analogies Metaphors?" In *Metaphor: Problems and Perspectives,* 1982

[8] Islam Md. Zahurul, *English to Bangla Phrase-Based Statistical Machine Translation,* Master thesis, Saarland University, 2009

[9] Joshi Yoga Raj, *A Chunk Alignment Model for Statistical Machine Translation on English-Nepali Parallel Corpus,* Master Thesis, Tribhuvan University, 2009

[10] Jurafsky Daniel, Martin James H., Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition (Prentice hall, Second Edition)

[11] Khadka Dinesh Kumar, *An Online Dictionary Approach to Handle Unknown Words in Machine Translation (English language to Nepali Language Translation),* Master Thesis, Tribhuvan University, 2011

[12] Kister Ken, Dictionaries defined. *Library Journal*, Vol. 117 Issue 11, 1992

[13] Langlais Philippe and Patry Alexandre, *Translating Unknown Words by Analogical Learning,* 2007

[14] Langlais Philippe, Yvon François and Zweigenbaum Pierre, *Improvements in Analogical Learning: Application to Translating multi-terms of the Medical Domain*, 2008

[15] Langlais Philippe, Yvon François, *Scaling up Analogical Learning*, 2008

[16] Lepage Yves, *Solving Analogies on Words: An Algorithm*, 1998

[17] Nicolas Stroppa & Yvon Francois, *An Analogical Learner for Morphological Analysis*, 2005

[18] Nizar Habash, *Four Techniques for Online Handling of Out-of-vocabulary Words in Arabic-English Statistical Machine Translation*, 2008

[19] Somers Harold, Dandapat Sandipan, Naskar Sudip Kumar, *A review of EBMT using proportional analogies*, 2009