Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Punjabi DeConverter for generating Punjabi from Universal Networking Language

Parteek KUMAR[1], Rajendra Kumar SHARMA[2]
([1]*Department of Computer Science & Engineering, Thapar University, Patiala 147004, India*)
([2]*School of Mathematics & Computer Applications, Thapar University, Patiala 147004, India*)
E-mail: {parteek.bhatia, rksharma}@thapar.edu

Received Mar. 11, 2012; Revision accepted Jan. 13, 2013; Crosschecked Feb. 22, 2013

Abstract: DeConverter is core software in a Universal Networking Language (UNL) system. A UNL system has EnConverter and DeConverter as its two major components. EnConverter is used to convert a natural language sentence into an equivalent UNL expression, and DeConverter is used to generate a natural language sentence from an input UNL expression. This paper presents design and development of a Punjabi DeConverter. It describes five phases of the proposed Punjabi DeConverter, i.e., UNL parser, lexeme selection, morphology generation, function word insertion, and syntactic linearization. This paper also illustrates all these phases of the Punjabi DeConverter with a special focus on syntactic linearization issues of the Punjabi DeConverter. Syntactic linearization is the process of defining arrangements of words in generated output. The algorithms and pseudocodes for implementation of syntactic linearization of a simple UNL graph, a UNL graph with scope nodes and a node having un-traversed parents or multiple parents in a UNL graph have been discussed in this paper. Special cases of syntactic linearization with respect to Punjabi language for UNL relations like 'and', 'or', 'fmt', 'cnt', and 'seq' have also been presented in this paper. This paper also provides implementation results of the proposed Punjabi DeConverter. The DeConverter has been tested on 1000 UNL expressions by considering a Spanish UNL language server and agricultural domain threads developed by Indian Institute of Technology (IIT), Bombay, India, as gold-standards. The proposed system generates 89.0% grammatically correct sentences, 92.0% faithful sentences to the original sentences, and has a fluency score of 3.61 and an adequacy score of 3.70 on a 4-point scale. The system is also able to achieve a bilingual evaluation understudy (BLEU) score of 0.72.

Key words: DeConverter, EnConverter, Machine translation, Universal Networking Language (UNL), Syntactic linearization
doi:10.1631/jzus.C1200061 **Document code:** A **CLC number:** TP391

1 Introduction

In the age of information technology, Internet has become an ocean of information. A large portion of information is still beyond the reach of a significant portion of society, however, because most of the information is available in English. In this multilingual scenario, machine translation (MT) is considered as an important tool to empower society. Universal Networking Language (UNL) based MT (developed with an interlingua-based approach) is also an effort in this direction. UNL programme was launched in 1996 in Institute of Advanced Studies (IAS) of United Nations University (UNU), Tokyo, Japan, and it is currently supported by the Universal Networking Digital Language (UNDL) Foundation, an autonomous organization. The approach in UNL revolves around the development of the EnConverter and the DeConverter for a natural language. EnConverter is used to convert a given sentence in natural language to an equivalent UNL expression, and DeConverter is used to convert a given UNL expression to an equivalent natural language sentence. A UNL system has the potential to bridge language barriers across the world with development of $2n$ components, while traditional approaches require $n(n-1)$ components, where n is the number of languages (Raman and Alwar, 1990).

In this paper, design and development of a Punjabi DeConverter has been presented. This paper highlights syntactic linearization issues of the Punjabi DeConverter. Syntactic linearization is the process of defining arrangements of words in generated output. This phase plays a vital role in the quality of the generation process. Algorithms and pseudocodes for implementation of the syntactic linearization phase have been discussed for the proposed Punjabi DeConverter.

Punjabi language is an Indo-Aryan language and one of the constitutionally recognized languages of India. Punjabi is widely spoken in north-west India, Pakistan, US, Australia, UK, and Canada. There are more than 91 million native speakers of Punjabi language, which makes it approximately the 12th most widely spoken language in the world (Lewis, 2009).

2 Universal Networking Language (UNL) system and its structure

In the UNL framework, two systems, EnConverter and DeConverter, need to be developed for a given natural language. The process of converting a source language (natural language) expression into a UNL expression is referred to as EnConversion, and the process of converting a UNL expression into a target language (natural language) expression is called DeConversion. The EnConverter and DeConverter for a language form a language server that may reside on the Internet. Both the EnConverter and the DeConverter perform their functions on the basis of a set of grammar rules and a word dictionary of a given language (Boguslavsky *et al*., 2005; http://www.undl.org). UNL represents information at sentence level in the form of universal words (UWs), UNL relations, and UNL attributes. The concepts represented by UWs and UNL relations are used to specify the role of each word in a sentence. Subjective meaning of a sentence is expressed through UNL attributes (Uchida, 2005).

Let us consider an example sentence (1) to illustrate UNL expression:

The boy is eating rice with a spoon. (1)

The UNL expression for sentence (1) is as follows:

```
{unl}
agt(eat(icl>do).@entry.@present.@progress,
    boy(icl>male person))
obj(eat(icl>do).@entry.@present.@progress, rice)
ins(eat(icl>do).@entry.@present.@progress,
    spoon)
{/unl}                                        (2)
```

The UNL graph for UNL expression (2) is as shown in Fig. 1. Here, ‘agt’ is a UNL relation which indicates ‘a thing that initiates an action’, ‘obj’ is a UNL relation that indicates ‘a thing in focus which is directly affected by an event’, ‘ins’ is a UNL relation used to indicate ‘an instrument to carry out an event’, and ‘@entry.@present.@progress’ are UNL attributes which indicate the main verb predicate and tense information of a given sentence.

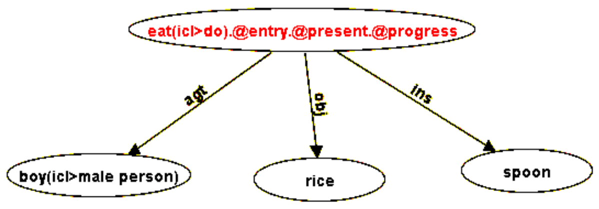

Fig. 1 Universal Networking Language (UNL) graph for UNL expression (2)

3 Related works

UNDL Foundation has provided EnConversion tool ‘EnCo’ and DeConversion tool ‘DeCo’ for automatic EnConversion and DeConversion processes, respectively. Martins *et al.* (1997) have used the DeCo tool for DeConverting UNL expressions into Brazilian Portuguese. Boguslavsky *et al.* (2000) have presented a multi-functional environment ‘ETAP-3’, an extension of the ‘ETAP’ MT system. The UNL module implemented in ETAP-3 naturally combines the transfer-based approach and interlingua approach. In their work, they have also proposed the architecture of a UNL-Russian DeConverter. Multilingual information processing through UNL has been proposed by Bhattacharyya (2001) and Dave *et al.* (2001). They proposed DeConverter for Hindi and Marathi with the use of the DeCo tool. Dhanabalan and Geetha (2003) have proposed a DeCo tool based UNL to Tamil DeConverter. Blanc (2005) has performed integration of an exiting MT system 'ARIANE-G5' to a proposed French DeConverter. Shi and Chen (2005) have proposed a UNL DeConverter for Chinese language. They have highlighted the problems of DeCo tool provided by the UNDL center, which include difficulty in writing rules, slow speed, and non-availability of source codes. These issues motivated them to propose a new DeConverter for Chinese. Pelizzoni and Nunes (2005) have introduced the 'Manati' DeConversion model as a UNL mediated Portuguese-Brazilian sign language human-aided MT system. Daoud (2005) has proposed an Arabic DeConversion system which involves mapping of relations, lexical transfer, word ordering, and morphological generations. Keshari and Bista (2005) have proposed architecture and design of the UNL Nepali DeConverter for the DeCo tool. The proposed system has two major modules, syntactic linearization module and morphology generation module. Singh *et al.* (2007) have proposed a DeConverter for Hindi language known as 'HinD', indicating non-availability of source codes of the DeCo tool and its complex rule-format. Their system consists of four main stages, including lexeme selection, morphological generation of lexical words, function word insertion, and syntactic linearization. All these components use language-independent algorithms operating on language-dependent data.

The issues brought forward by researchers about the DeCo tool have motivated us to work on a UNL-Punjabi DeConverter. As such, a new DeConverter has been implemented for DeConversion of UNL expressions to Punjabi language sentences in this work.

4 Architecture of UNL-Punjabi DeConverter

4.1 Architecture of Punjabi DeConverter

A general architecture of Punjabi DeConverter is given in Fig. 2. It makes use of language-independent and language-dependent components during the generation process. The first stage of DeConverter is the UNL parser which parses an input UNL expression and builds a node-net from the input UNL expression. The node-net has a directed acyclic graph (DAG) structure. During the lexeme selection stage, target language (i.e., Punjabi) root words and their dictionary attributes are selected for given UWs in the input UNL expression from the Punjabi-UW dictionary. After that, nodes are ready for generation of morphology according to the target language in the morphology phase. In this stage, the root words may be changed; i.e., something can be added or removed to obtain the complete sense of words. The system makes use of morphology rules for this purpose. In the function word insertion phase, function words or case markers, such as ਨੇ nē, 'ਦੇ ਨਾਲ਼' 'dē nāl', ਨੂੰ nūṃ, ਤੋਂ tōṃ, ਦੇ ਲਈ dē laī, ਵਾਸਤੇ vāsatē, ਦਾ dā, ਦੇ dē, ਦੀ dī, are inserted to morphed words. These function words are inserted in a generated sentence, based on the rule base designed for this purpose. Finally, the syntactic linearization phase is used to define word order in the generated sentence so that the output matches a natural language sentence (Nalawade, 2007; Singh *et al.*, 2007).

Working of Punjabi DeConverter is illustrated with an example sentence given in (3).

Punjabi sentence:

ਮੁੰਡੇ ਨੇ ਬਾਗ਼ ਵਿਚ ਫ਼ੁੱਟਬਾਲ ਖੇਡਿਆ ।

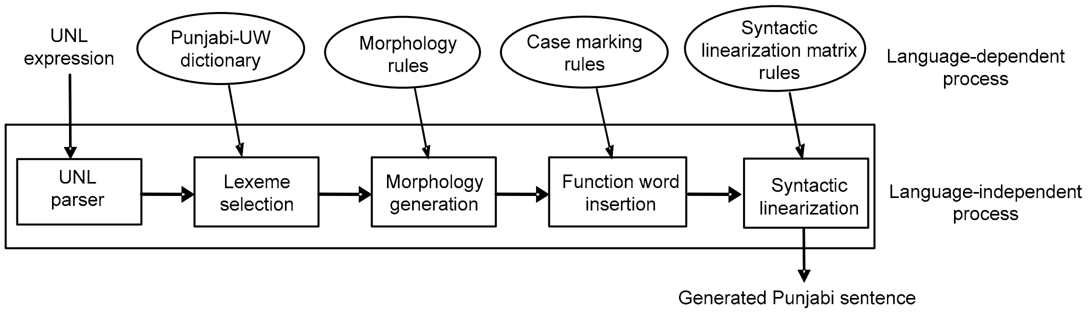

Fig. 2 Architecture of Universal Networking Language (UNL)-Punjabi DeConverter

Transliterated sentence:

muṇḍē nē bāġ vic fuṭṭbāl khēḍiā.

Equivalent English sentence:

The boy played football in the garden. (3)

The UNL expression for example sentence (3) is given in (4).

```
{unl}
agt(play(agt>human,obj>game).@past.@entry,
    boy(icl>male person))
obj(play(agt>human,obj>game).@past.@entry,
    football(icl>game))
plc(play(agt>human,obj>game).@past.@entry,
    garden(icl>place))
{/unl}                                        (4)
```

To convert UNL expression (4) to the natural language Punjabi sentence, Punjabi DeConverter is used.

The UNL expression acts as input for the Punjabi DeConverter. The UNL parser checks the input UNL expression for errors and generates the node-net or UNL graph as depicted in Fig. 3.

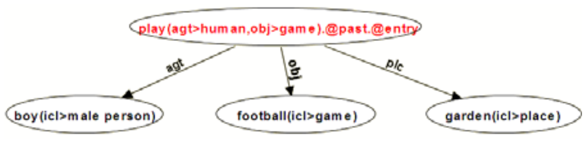

Fig. 3 UNL graph generated by the UNL parser for UNL expression (4)

The Lexeme selection phase populates the node-list with equivalent Punjabi words for the UWs given in the input UNL expression. The populated node-list is given in (5).

Node$_1$: Punjabi word: ਖੇਡ khēḍ;

UW: play(agt>human, obj>game).@past.@entry

Node$_2$: Punjabi word: ਮੁੰਡਾ muṇḍā;

UW: boy(icl>male person)

Node$_3$: Punjabi word: ਫੁੱਟਬਾਲ fuṭṭbāl;

UW: football(icl>game)

Node$_4$: Punjabi word: ਬਾਗ਼ bāġ;

UW: garden(icl>place) (5)

In the morphology phase, morphological rules are applied to modify Punjabi words stored in the nodes according to UNL attributes given in the input UNL expression and dictionary attributes retrieved from the Punjabi-UW dictionary. The nodes given in (5) are processed by the morphology rules. The processed nodes are given in (6).

Node$_1$: Punjabi word: ਖੇਡਿਆ khēḍiā;

UW: play(agt>human, obj>game).@past.@entry

Node$_2$: Punjabi word: ਮੁੰਡੇ muṇḍē;

UW: boy(icl>male person)

Node$_3$: Punjabi word: ਫੁੱਟਬਾਲ fuṭṭbāl;

UW: football(icl>game)

Node$_4$: Punjabi word: ਬਾਗ਼ bāġ;

UW: garden(icl>place) (6)

It is evident from nodes given in (6) that, in the morphology phase ਖੇਡ khēḍ 'play' is changed to ਖੇਡਿਆ khēḍiā 'played' and ਮੁੰਡਾ muṇḍā 'boy' is changed to ਮੁੰਡੇ muṇḍē by morphology rules. The function word insertion phase inserts function words in the morphed lexicon. Nodes processed by the function word insertion phase are given in (7).

Node$_1$: Punjabi word: ਖੇਡਿਆ khēḍiā;

UW: play(agt>human, obj>game).@past.@entry

Node$_2$: Punjabi word: ਮੁੰਡੇ ਨੇ muṇḍē nē;

UW: boy(icl>male person)

Node$_3$: Punjabi word: ਫੁੱਟਬਾਲ fuṭṭbāl;

UW: football(icl>game)

Node$_4$: Punjabi word: ਬਾਗ਼ ਵਿਚ bāġ vic;

UW: garden(icl>place) (7)

In this phase, case markers ਨੇ nē and ਵਿਚ vic 'in' are added to Node$_2$ and Node$_4$, respectively, according to the function word insertion rule base. In the syntactic linearization phase, one traverses the nodes given in (7) in a specific sequence based on the syntactic linearization rule base for Punjabi language.

The sequence for processing of nodes is given in (8) and the Punjabi sentence generated by this sequence is given in (9).

$$\text{Node}_2\ \text{Node}_4\ \text{Node}_3\ \text{Node}_1 \tag{8}$$

ਮੁੰਡੇ ਨੇ ਬਾਗ਼ ਵਿਚ ਫ਼ੁੱਟਬਾਲ ਖੇਡਿਆ । (9)

muṇḍē nē bāġ vicfuṭṭbāl khēḍiā.

It is evident from the generated Punjabi sentence (9) that the system is able to convert an input UNL expression to Punjabi successfully.

The descriptions of different phases of the Punjabi DeConverter are given in the next.

4.2 Phases of Pubjabi DeConverter

4.2.1 UNL parser phase

The UNL parser is the first phase of UNL-Punjabi DeConverter. It is used to parse an input UNL expression to report the errors if any in the input expression. If the input expression is in a proper format or free from errors, then it builds a semantic net known as node-net structure for the input UNL expression. This node-net is commonly called a UNL graph. A UNL graph consists of nodes and edges. A node in the UNL graph represents a concept in the form of UW. An edge in the node-net represents a UNL binary relation between two nodes. The edges in a UNL graph are directed from the parent node to child node. The system also maintains the access path from child to its parent for the purpose of backtracking.

4.2.2 Lexeme selection phase

Lexeme selection is the process of selecting target language words for UWs given in the input UNL expression. During lexeme selection, UWs are searched in dictionary along with constraints specified in the input UNL expression. This phase uses a Punjabi-UW dictionary for this task. A Punjabi-UW dictionary containing 115 000 entries has been developed by considering the Hindi-UW dictionary (Indian Institute of Technology (IIT), Bombay, India) as a reference. While developing the Punjabi-UW dictionary from the Hindi-UW dictionary, language-independent components have not been changed but language-dependent components, such as vowel-ending and morphology information, are changed as per Punjabi language. The Hindi headwords are also replaced by equivalent Punjabi headwords during this process. As such, the Punjabi-UW dictionary consists of the Punjabi root word as the headword, a UW and a set of morphological, syntactic, and semantic attributes as its entries.

4.2.3 Morphology generation phase

In this phase, headwords are modified according to the morphology of target language. The system makes use of generation rules during this process. These generation rules are designed on the basis of analysis of Punjabi morphology carried out for this purpose. There are three categories of morphology that have been identified for the purpose of conversion of a UNL expression to equivalent Punjabi language sentences (Vachhani, 2006). They are attribute label resolution morphology, relation label resolution morphology, and noun, adjective, pronoun, and verb morphology.

Attribute label resolution morphology deals with generation of Punjabi words on the basis of UNL attributes attached to a node and its grammatical attributes retrieved from lexicon. The root words retrieved from Punjabi-UW dictionary are changed in this phase depending on their gender, number, person, tense, aspect, modality (GNPTAM), and vowel-ending information.

Relation label resolution morphology manages the prepositions in English or postpositions in Punjabi, because prepositions in English are similar to postpositions in Punjabi. These link noun, pronoun, and phrases to other parts of the sentence. Some Punjabi postpositions are ਨੇ nē, ਨੂੰ nūṃ, ਉੱਤੇ uttē 'over', ਦਾ dā 'of', ਕੋਲੋਂ kōlōṃ 'from', ਨੇੜੇ nēḍaē 'near', ਲਾਗੇ lāgē 'near', etc. Insertion of these words in generated output depends upon the information encoded in UNL relations of a given UNL expression. In relation label morphology, most UNL relation labels introduce postpositions (also known as function words or case markers) between the child node and parent node during the generation process. Generation of these words depends upon UNL relation and the conditions imposed on parent and child nodes' attributes of UNL relation. Relation label morphology is used to prepare the rule base used during the function word insertion phase of DeConverter as discussed in the following.

With attribute and relation label morphology, the system is able to generate a sentence very close to its natural form. The phonetic properties of a language are handled by noun, adjective, pronoun, and verb morphology of the DeConverter.

4.2.4 Function word insertion phase

The function word insertion phase is used to insert function words like case markers or postpositions and conjunctions in Punjabi (ਨੇ nē, ਨੂੰ nūṃ, ਉੱਤੇ uttē 'over', ਦਾ dā 'of', ਕੋਲੋਂ kōlōṃ 'from', ਅਤੇ 'and', etc.) to the morphed words generated at the morphology phase. Insertion of function words in generated output depends upon UNL relation and conditions imposed on parent and child nodes' attributes in a relation (Singh *et al.*, 2007). A rule base has been prepared for this purpose. For each of 46 UNL relations, different function words are used depending upon grammatical details of a target language (Dey and Bhattacharyya, 2005).

Following Sinha (2005) and Vachhani (2006), a rule base for insertion of the function word has been prepared. This rule base consists of nine columns. The description of each column of this rule format is given below.

1. First column: relation name

The name of UNL relation corresponding to which the reference to the rule base is being made is stored in this column.

2. Second column: the function word preceding the parent node

The function word, which should be inserted before a parent node of the relation in generated output, is stored in this column.

3. Third column: the function word following the parent node

The function word, which should be inserted after a parent node of a given relation in generated output, is stored in this column.

4. Fourth column: the function word preceding the child node

The function word, which should be inserted before a child node of the relation in generated output, is stored in this column.

5. Fifth column: the function word following the child node

The function word, which should be inserted after a child node of the relation in generated output, is stored in this column.

6. Sixth column: positive conditions for the parent node

The attributes whose presence needs to be asserted on the parent node for firing of the rule are stored in this column.

7. Seventh column: negative conditions for the parent node

The attributes whose absence needs to be asserted on the parent node for firing of the rule are stored here.

8. Eighth column: positive conditions for the child node

The attributes whose presence needs to be asserted on the child node for firing of the rule are stored in this column.

9. Ninth column: negative conditions for the child node

The attributes whose absence needs to be asserted on the child node for firing of the rule are stored.

If there is more than one attribute that needs to be asserted on a given node for firing of a rule, then they are stored in the rule base with the separation of '#' sign. Here, attributes represent UNL attributes (obtained from a given UNL expression) or lexical attributes (obtained from the Punjabi-UW dictionary) of a node.

The rule base for function word insertion is illustrated with an example rule given in (10).

agt:null:null:null::@past#V:VINT#
@progress#jA:N#3rd:1st#2nd, (10)

where 'agt' is a UNL relation under consideration, and firing of the given rule will result into insertion of function word ਨੇ nē following the child node in generated output, because the function word appears in the fifth column and the second, third, and fourth columns contain 'null' in the rule. The sixth column contains '@past#V', which means that the rule will be fired if the parent of 'agt' relation contains '@past' as its UNL attribute in the given input UNL expression and has a 'V' as its lexical attribute in Punjabi-UW dictionary. The seventh column contains 'VINT#@progress#jA' which refers to the attributes whose absence needs to be asserted on the parent node for firing of the rule. It means that the parent node should not contain 'VINT' (intransitive verb), 'jA' ('go' verb) attributes in the lexicon or the '@progress' attribute in the parent of UNL expression. The eighth column of the rule given in (10) contains 'N#3rd' which refers to the attribute whose presence needs to be asserted on the child node for firing of the rule; i.e., the child should have an 'N' (noun) and '3rd' (third person) attribute in the Punjabi-UW dictionary. The ninth column contains '1st# 2nd' which refers to the attribute whose absence needs to be asserted on the child node for firing of the rule. It means that the child node should not refer to the first person or the second person in the sentence. Thus, if the relation 'agt' has a parent node with an '@past' and 'V' attribute, without 'VINT', 'jA', '@progress', or '@custom' attribute, or has a child node with an 'N' and '3rd' attribute and without a '1st' or '2nd' attribute, then function word ਨੇ nē will be inserted following the child node in generated output.

For example, in UNL relation 'agt(play(agt>human, obj>game).@past.@entry, boy(icl>male person))' of UNL expression (2), the parent node of relation 'agt' is 'play(agt>human, obj>game)' having 'V' and '@past' attribute and without the 'VINT' attribute in the lexicon. The child node of 'agt' relation is 'boy(icl>male child)' that has 'N' and '3rd' attribute and does not have '1st' and '2nd' attributes in the lexicon. As such, this will result into the firing of rule (10) and thus the generation of function word ਨੇ nē followed by child node 'boy(icl> male child)' in generated output.

4.2.5 Syntactic linearization phase

Syntactic linearization is the process of linearizing the lexemes in the semantic hyper-graph. As such, it is a process to define the word order in the generated sentence. In a language, some word orders are considered more natural than others. Syntactic linearization deals with the arrangements of words in generated output so that output matches the natural language sentence. The system assigns relative positions to various words based on the relations they share with the headword in a sentence (Vachhani, 2006). The structural differences between English (subject-verb-object) and Punjabi (subject-object-verb, SOV) languages necessitate the syntactic linearization phase in the development of a Punjabi Deonverter.

5 Major issues in syntactic linearization

Parent-child relationship and matrix-based priority of relations are two important issues in the syntactic linearization phase.

5.1 Parent-child relationship

In a UNL binary relation rel(UW1, UW2), UW1 acts as the parent of the relation, whereas UW2 acts as the child of the relation. For each parent-child relationship, the system should state whether the parent should be ordered before or after the child in the generated output (Vora, 2002). For the syntactic linearization of Punjabi language, in most of UNL relations the parent node appears right to all of its children in the generated output.

To illustrate this concept, let us consider a UNL relation agt(UW1, UW2), where UW1 is a verb and UW2 is the subject or agent of that event. Since Punjabi is an SOV language, subject always comes to the left of the verb. Same is the case of 'obj' relation. In case of a UNL expression given in (11), both children, i.e., 'boy(icl>male child)' and 'rice(icl>food)', will be placed left to the parent 'eat'.

```
{unl}
agt(eat.@present.@entry, boy(icl>male child))
obj(eat.@present.@entry, rice(icl>food))
{/unl}                                          (11)
```

Since both children are inserted to the left of the parent, the child to be inserted first in generated output will be decided by the matrix-based priority of relations.

5.2 Matrix-based priority of relations

Necessity of matrix-based priority of relations occurs when the children of two or more UNL relations have a common parent. It is important to decide the relative positions of children (sharing a common parent) with respect to each other in generated output. Following Vora (2002), relative positions of children with respect to each other in the proposed Punjabi

DeConverter are decided using a matrix $\boldsymbol{M}$. $\boldsymbol{M}$ has 46 rows and 46 columns, representing 46 UNL relations specified in UNL specifications (Uchida, 2005). This matrix, $\boldsymbol{M}=[m_{ij}]$, where i=1, 2, …, 46 and j=1, 2, …, 46, contains the elements as 'L', 'R', and '-', where 'L' means towards left, 'R' means towards right, and '-' means no action.

If 'm_{ij}'='L', then the position of the child of the ith relation is left to the child of the jth relation when the two children share a common parent. If 'm_{ij}'='R', then the position of the child of the ith relation is right to the child of the jth relation when the two children share a common parent. If 'm_{ij}'='-', then no action is to be taken as it is impossible that the child of this ith relation shares a common parent with the child of this jth relation (Hrushikesh, 2002; Vachhani, 2006). This is illustrated with 'R_i' and 'R_j' as two UNL binary relations between three nodes 'N_1', 'N_2', and 'N_3' as $R_i(N_3, N_1)$ and $R_j(N_3, N_2)$. Here, nodes 'N_1' and 'N_2' are the children of the same parent 'N_3' (Fig. 4).

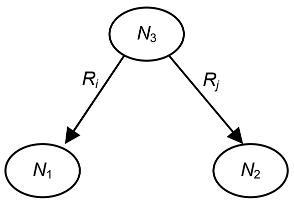

Fig. 4 UNL graph of two nodes with the same parent

Based on the structure of target language, if 'N_1' appears at the left of 'N_2' in the generated sentence as denoted by '(N_1 L N_2)', then the priority matrix shown in Fig. 5 needs to be maintained for its syntactic linearization.

	R_i	R_j
R_i	-	L
R_j	R	-

Fig. 5 Matrix representation for (N_1 L N_2) structure

Priority of the child of a relation depends upon the frequency of 'L' in its row. If a relation has all 'L' in its row, then the child node of that relation will have the highest priority and it will appear at extreme left in generated output. Similarly, if a relation has all 'R' in its row, then the child node of that relation will have the lowest priority and will appear at extreme right of all the children sharing a common parent in generated output (Vachhani, 2006; Nalawade, 2007). Owing to the fact that we can associate a priority to the child associated with a relation, $\boldsymbol{M}$ is called a priority matrix.

6 Syntactic linearization for simple sentences

A simple sentence contains a subject and a verb, and it expresses a complete thought. The UNL expression of simple sentences is converted into a simple node-net or UNL graph by the UNL parser. The processing sequence of nodes of this UNL graph controls the word order in the generated output. A program has been developed in Java to control the sequence of processing of nodes of a simple UNL graph. PseudoCode 1 (Table 1) contains the instructions corresponding to this program.

PseudoCode 1 performs processing of nodes of the UNL graph according to the priority matrix. The given PseudoCode works in a loop, and control exits out of it after the processing of all nodes of the graph. It first traverses the highest priority child node sharing a common parent. If a node has no further unvisited child, then it is processed and added into the final string. After processing of the child node, the parent of that node becomes an active node and again traverses the highest priority un-processed child node. This process continues until the entry node gets processed.

The working of PseudoCode 1 is illustrated with an example English sentence given in (12).

He has scored 80% marks. (12)

The UNL expression for this example sentence is

```
{unl}
agt(score.@present.@complete.@entry, he)
obj(score.@present.@complete.@entry, marks)
mod(marks, percent)
qua(percent, 80)
{/unl}                                        (13)
```

The UNL graph for UNL expression (13) is depicted in Fig. 6.

Table 1 PseudoCode 1 to control the processing sequence of nodes of simple UNL graph

```
1   Begin
2       start traversing the graph from the entry node and set this as the active node;
3       while (true)
4           if (active node has no parent)
5               if (active node has no unprocessed child)
6                   add node to final string;
7                   Mark node as processed;
8                   exit from the loop;
9               Else
10                  if (node is not already marked as visited)
11                      mark the node as a visited node;
12                  end-if
13                  get the highest priority unprocessed child relation;
14                  set the highest priority unprocessed child relation as an active node;
15              end-if
16          else
17              if (node has one parent)
18                  if (active node has no unvisited child)
19                      add node to final string;
20                      mark the node as processed;
21                      set the parent of the node as an active node;
22                  else
23                      if (node is not already marked as visited)
24                          mark the node as a visited node;
25                      end-if
26                      get the highest priority unprocessed relation child;
27                      set the highest priority unprocessed child relation as the active node;
28                  end-if
29              end-if
30          end-if
31      end-while
32  End
```

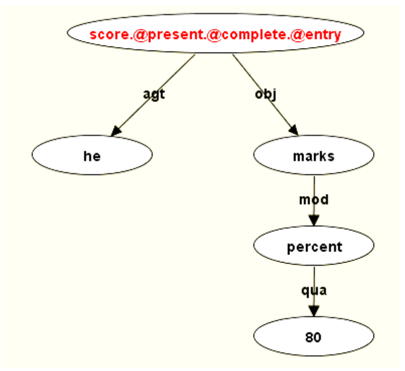

Fig. 6 UNL graph for the UNL expression (13)

The node-list for the UNL expression given in (13) after processing of the morphology phase and function word insertion phase is given in (14).

Node$_1$: Punjabi word: ਹਾਸਲ ਕੀਤੇ hāsal kītē;
UW: score
Node$_2$: Punjabi word: ਉਸ ਨੇ us nē;
UW: he
Node$_3$: Punjabi word: ਅੰਕ aṅk;
UW: marks
Node$_4$: Punjabi word: ਫ਼ੀਸਦੀ fīsadī;
UW: percent
Node$_5$: Punjabi word: 80;
UW: 80 (14)

In Fig. 6, the entry node is 'score.@present.@complete.@entry' as this contains '@entry' attribute, and the labels on edges indicate UNL relation labels. Here, priority of ‘agt’ is higher than the priority of ‘obj’ relation as shown in Fig. 6. Thus, the ‘he’ node will be traversed first. Note that the priority matrix given in Fig. 7 is a sub-matrix of earlier defined matrix ***M***.

	agt	obj
agt	-	*L*
obj	*R*	-

Priorities of relations
agt: 1
obj: 0

Fig. 7 Priority matrix for ‘agt’ and ‘obj’ relations

The ‘he’ node has no child and its parent node ‘score’ has already visited, so it will be processed and its Punjabi word attribute will be appended to the final string used to store generated output; i.e., the final string will become ‘ਉਸ ਨੇ’. Now, the parent of the ‘he’ node, i.e., the ‘score’ node, will become an active node. It has only one unprocessed child, i.e., the ‘marks’ node. Thus, it will be traversed next and marked as visited. It has one unprocessed child node ‘percent’, so it will be traversed and marked as visited. The ‘percent’ node also has one unprocessed child node ‘80’, so it will be traversed next and marked as visited. The node ‘80’ has no further unprocessed child, so it will finally be processed and its Punjabi word attribute will be appended to the final string; i.e., the final string will become ‘ਉਸ ਨੇ 80’. The parent of the ‘80’ node, i.e., the ‘percent’ node, now will become an active node. The ‘percent’ node has no unprocessed child, so it will be processed and its Punjabi word attribute will be appended to the final string; i.e., the final string will become ‘ਉਸ ਨੇ 80 ਫ਼ੀਸਦੀ’. Now, the parent of the ‘percent’ node, i.e., the ‘marks’ node, will become an active node. The ‘marks’ node has no unprocessed child, so it will be processed and its Punjabi word attribute will be appended to the final string; i.e., the final string will become ‘ਉਸ ਨੇ 80 ਫ਼ੀਸਦੀ ਅੰਕ’. Now, the parent of the ‘marks’ node, i.e., the ‘score’ node, will become an active node. It is an entry node and has no unprocessed child, so it will be processed and its Punjabi word will be appended to the final string; i.e., the final string will become ‘ਉਸ ਨੇ 80 ਫ਼ੀਸਦੀ ਅੰਕ ਹਾਸਲ ਕੀਤੇ’. Since the entry node is processed, control will exit from loop and final output according to the syntactic linearization will be available in the final string. Thus, PseudoCode 1 will result in the processing of the node-net in the order given in (15) and it will result into a Punjabi sentence shown in (16) as generated output of Punjabi De-Converter.

$$\text{Node}_2\ \text{Node}_5\ \text{Node}_4\ \text{Node}_3\ \text{Node}_1, \qquad (15)$$

ਉਸ ਨੇ 80 ਫ਼ੀਸਦੀ ਅੰਕ ਹਾਸਲ ਕੀਤੇ । (16)

us nē 80 fīsadī aṅk hāsal kītē.

The equivalent English sentence:

He has scored 80% marks.

7 Syntactic linearization of a UNL graph with a scope node

Scope is used to represent a compound universal word or a compound concept. A compound concept is a set of binary relations that are grouped together to express a complex concept. This is defined by adding a compound universal word identifier (UW-ID) immediately after the relation label. A compound UW is referred to with its ID instead of a universal word.

The syntactic linearization for sentences with a compound concept is a bit different from that for simple sentences. In this case, a UNL graph contains a scope node which itself is a UNL graph (Fig. 8).

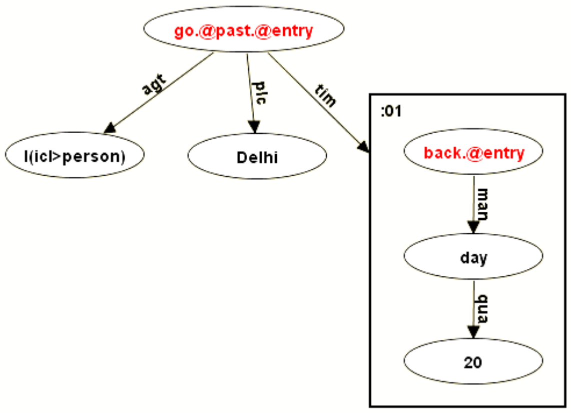

Fig. 8 UNL graph with a scope node for the UNL expression given in (18)

Algorithm 1 has been used to produce syntactic linearization for a UNL graph with a scope node.

Algorithm 1 (Processing the nodes of the UNL graph with a scope node)

1. Develop syntactic linearization by considering the scope node as a single node with PseudoCode 1.
2. Develop syntactic linearization of the scope node's UNL graph using PseudoCode 1.
3. Replace the scope node in the output generated in step 1 with the output generated in step 2 to obtain the final generated sentence.

Syntactic linearization of the UNL graph with a scope node is illustrated by an example sentence given in (17).

I went to Delhi 20 days back. (17)

The UNL expression of this example sentence is

```
{unl}
agt(go.@past.@entry, I(icl>person))
plc(go.@past.@entry, Delhi)
tim(go.@past.@entry, :01)
man:01(back.@entry, day)
qua:01(day, 20)
{/unl}                                    (18)
```

The UNL graph of this UNL expression is depicted in Fig. 8. In Fig. 8, 'go' is the entry node having three children, 'I(icl>person)' with 'agt' relation, 'Delhi' with 'plc' relation, and scope node ':01' with 'tim' relation. Here, 'agt' relation has the highest priority followed by 'tim' relation and then by 'plc' relation, as shown by the priority matrix in Fig. 9. Thus, the system will first traverse the 'I(icl>person)' node, followed by the scope node ':01', and then by the 'Delhi' node.

	agt	plc	tim
agt		*L*	*L*
plc	-	-	*R*
tim	*R*	*L*	-

Priorities of relations
agt: 2
plc: 0
tim: 1

Fig. 9 Priority matrix for 'agt', 'plc', and 'tim' relations

Using PseudoCode 1, while considering the scope node as a single node, syntactic linearization of this UNL graph by considering UWs (without constraints) shall be

I:01 Delhi go. (19)

Using Algorithm 1, the scope node ':01' shown in output (19) is replaced by the output generated from syntactic linearization of the scope node's UNL graph. Again, using PseudoCode 1, syntactic linearization of UWs (without constraints) of the scope node's UNL graph is

20 day back. (20)

The final output for the UNL expression given in (18) is generated by replacing the scope node in (19) with syntactic linearization of UWs given in (20). Thus, the final syntactic linearization of UWs (without constraints) for the UNL graph depicted in Fig. 8 will be generated as

I 20 day back Delhi go. (21)

A Punjabi sentence generated after applying the morphology, function word insertion, and syntactic linearization phases of the Punjabi DeConverter is given in (22).

ਮੈਂ 20 ਦਿਨ ਪਹਿਲਾਂ ਦਿੱਲੀ ਗਿਆ । (22)
maiṃ 20 din pahilāṃ dillī giā.

8 Untraversed parent handling

While traversing a UNL graph, the system may encounter a situation where the parent of a node is untraversed (Vachhani, 2006). This is illustrated with the help of an example expression given in (23).

Above mentioned DeConverter detail. (23)

The UNL expression for this example expression is given in (24), and the corresponding UNL graph is given in Fig. 10.

```
{unl}
obj(mention, detail.@entry)
plc(mention, above)
mod(detail.@entry, DeConveter)
{/unl}                                   (24)
```

In Fig. 10, the 'detail' node acts as the entry node. According to PseudoCode 1, traversal of the UNL graph starts from the entry node, i.e., the 'detail' node. Here, the system encounters a case of untraversed parent, because the parent of the 'entry' node, i.e., the 'mention' node, has not been traversed by the system. The following strategy has been implemented for handling an untraversed parent node.

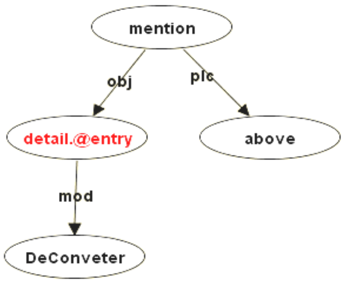

Fig. 10 UNL graph of UNL expression (24)

If the system encounters a node having an untraversed parent, then that node will be removed as a child of its parent and will be added as a virtual parent to its untraversed parent. The parent of that node will be set to 'null' and the untraversed parent node will be set as an active node. Subsequent syntactic linearization will be carried out according to PseudoCode 1. PseudoCode 2 has been used to implement this strategy.

PseudoCode 2 (Handling of the untraversed parent node)

```
if (parent of active node is untraversed)
   add the active node as a virtual parent to its untraversed parent;
   remove the active node as a child of its parent;
   set the active node parent to null;
   set the untraversed parent node as an active node;
end-if
```

PseudoCode 2 is inserted between lines numbered 17 and 18 of PseudoCode 1 to extend it to handle the UNL graph having an untraversed parent node.

According to PseudoCode 2, for processing of the UNL graph in Fig. 10, the system removes the 'detail' node as the child of its untraversed parent node 'mention' and sets its parent to null. Now, the system adds the 'detail' node as a virtual parent of its untraversed parent node 'mention'. The system then sets the untraversed parent node, i.e., the 'mention' node, as an active node. The modified UNL graph after these actions is depicted in Fig. 11.

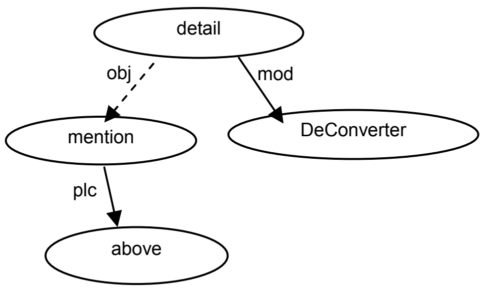

Fig. 11 Modified UNL graph for a node having an untraversed parent

After starting traversal of the UNL graph (Fig. 10) from the entry node, i.e., 'detail', the system encounters a situation of untraversed parent. The system modifies the UNL graph according to PseudoCode 2 and sets the 'mention' node as an active node. It has one untraversed child node, i.e., 'above'. The 'above' node has no further unprocessed child, so its Punjabi word attribute will be appended to the final string used to store the generated output. It has the parent node 'mention' which has already traversed and has no unprocessed child node; thus, it will be processed by the system and its Punjabi word attribute will be appended to the final string. The 'mention' node has one virtual parent, i.e., the 'detail' node; hence, it will become an active node. This node has one child node, i.e., 'DeConverter', so it will become an active node. It has no unprocessed child, so it will be processed and its Punjabi word attribute will be appended to the final string. The parent of the 'DeConverter' node, i.e., the 'detail' node, will now be set as an active node. It has no parent and no unprocessed child, so the 'detail' node is processed by the system and its Punjabi word attribute will be appended to the final string. Since all the nodes are processed by the system, control will exit and final output according to syntactic linearization will be available in the final string.

Output given in (25) indicates a final syntactic linearization of UWs (without constraints) and output given in (26) shows the generated Punjabi sentence after application of the morphology, function word insertion, and syntactic linearization phases of the Punjabi DeConverter.

above mention DeConverter detail, (25)
ਉੱਪਰ ਦਿੱਤੀ ਡੀਕੰਨਵਰਟਰ ਦੀ ਵਿਆਖਿਆ । (26)
uppar dittī dīknnavratar dī viākhiā.

9 Handling of multiple parents

Normally, in a UNL graph, each child node has only one parent; i.e., one node plays only a single role. But sometimes, one child may have more than one parent; i.e., one node plays more than one role in a UNL expression. In such a case, the system may encounter a situation where the parent of a node may be untraversed or unvisited. This situation is illustrated with an example sentence given in (27).

His eyes are affected by strong viral infection. (27)

The UNL expression for the example sentence given in (27) is

```
{unl}
obj(affect.@present.@entry, eye.@pl)
pos(eye.@pl, he)
agt(affect.@present.@entry, infection)
aoj(viral, infection)
mod(infection, strong)
{/unl}                                        (28)
```

The UNL graph of the UNL expression given in (28) is depicted in Fig. 12. Here, the entry node 'affect' has two children 'eye' and 'infection' associated with UNL relations 'obj' and 'agt', respectively. The 'agt' relation is of the highest priority, so the 'infection' node will be traversed first by the system. Now, the system encounters two parents for an active node 'infection'. The following strategy has been implemented for handling multiple parents.

If the system encounters a node having multiple parents, then its untraversed parent will be detected by the system and the active node will be set as a virtual parent of the untraversed parent. The untraversed parent will now be removed as the parent of the active node, and the active node will be removed as a child of the untraversed parent. The untraversed parent node will be set as the active node and further processing of syntactic linearization will be carried out according to PseudoCode 1. PseudoCode 3 has been used to implement this strategy.

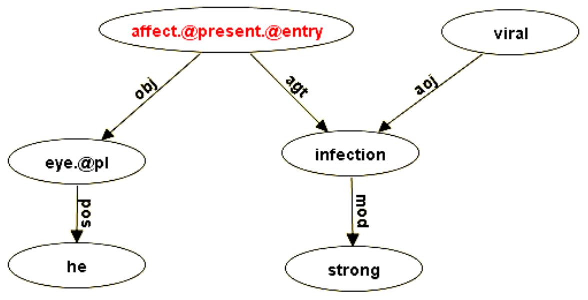

Fig. 12 UNL graph having multiple parents for the UNL expression (28)

PseudoCode 3 (Handling nodes with multiple parents nodes)

```
if (node has multiple parents)
  find an untraversed parent;
  set the active node as a virtual parent of its untraversed parent;
  remove the untraversed parent from the active node's parent list;
  remove the active node as a child of its untraversed parent;
  set the untraversed parent node as an active node;
end-if
```

PseudoCode 3 is inserted between lines numbered 29 and 30 of PseudoCode 1 for extending it to handle the UNL graphs having nodes with multiple parents.

According to PseudoCode 3, for processing of the UNL graph depicted in Fig. 12, the system will find the untraversed parent of the active node having multiple parents, i.e., 'infection'. The node 'viral' will be identified as the untraversed parent node of the active node. According to PseudoCode 3, the system will set 'infection' as a virtual parent of the untraversed parent node, i.e., 'viral' (Fig. 13).

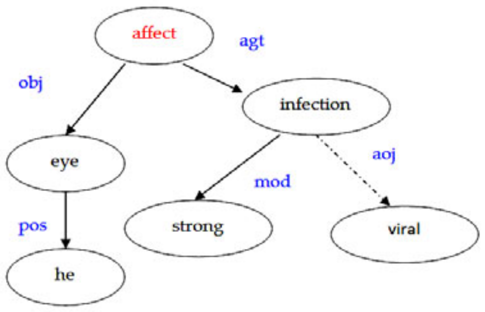

Fig. 13 Modified UNL graph for a node having multiple parents

Output given in (29) contains a final syntactic linearization of UWs (without constraints) and the output given in (30) contains a generated Punjabi sentence after application of the morphology, function word insertion, and syntactic linearization phases of the Punjabi DeConverter.

strong viral infection he eye affect, (29)

ਤਕੜੇ ਜੀਵਾਣਿਕ ਡੂਤ ਨੇ ਉਸ ਦੀਆਂ ਅੱਖਾਂ ਨੂੰ ਅਸਰ ਕੀਤਾ। (30)

takṛē jīvāṇik saṇkramaṇ nē us dīāṃ akkhāṃ nūṃ asar kītā.

10 Special cases in syntactic linearization

Vachhani (2006) and Nalawade (2007) have identified some UNL relations that need to be treated as special cases during their syntactic linearization for Hindi language. It has been explored that these cases are also applicable for Punjabi language. These UNL relations are: ‘and’ and/or ‘or’ relation(s), ‘fmt’ relation, ‘cnt’ relation, and ‘seq’ relation. The following strategies have been formulated for syntactic linearization of these UNL relations.

10.1 Strategy for ‘and’ and/or ‘or’ relation(s)

For syntactic linearization of the UNL graph having the ‘and’ and/or ‘or’ relation(s), PseudoCode 1 requires a minor modification. While traversing a UNL graph, if the system encounters the relation ‘and’ or ‘or’ as the highest priority relation, then the parent node of the relation ‘and’ or ‘or’ will be processed first and its child node will be set as an active node. Further syntactic linearization will be carried out according to PseudoCode 1. This strategy has been implemented in PseudoCode 4.

PseudoCode 4 (Handling of ‘and’ and/or ‘or’ relations)

```
if (highest priority relation is ‘and’ or ‘or’)
    process the active node by appending its Punjabi
        word attribute into the final string;
    mark the node as a special node to indicate that it
        has already processed;
end-if
```

PseudoCode 4 is inserted between lines 13 and 14, and also between lines 26 and 27 of PseudoCode 1 for extending it to handle the UNL graphs having ‘and’ and/or ‘or’ relations.

10.2 Strategy for ‘fmt’ relation

Similarly, if during syntactic linearization of a UNL graph, the system encounters the relation ‘fmt’ as the highest priority relation, the parent node of relation ‘fmt’ is processed first, followed by the processing of its child node. After the processing of the child node, the child node will be set as an active node. Further processing will be carried out according to PseudoCode 1. This strategy has been implemented in PseudoCode 5.

PseudoCode 5 (Handling of the ‘fmt’ relation)

```
if (highest priority relation is ‘fmt’)
    process the active node by appending its Punjabi
        word attribute into the final string;
    mark the node as a special node to indicate that it
        has already processed;
    process its child node by appending its Punjabi
        word attribute into the final string;
    mark the child node as a special node to indicate
        that it has already processed;
    set the child node as the active node;
end-if
```

PseudoCode 5 is inserted between lines 13 and 14, and also between 26 and 27 of PseudoCode 1 for extending it to handle the UNL graphs having ‘fmt’ relation.

10.3 Strategy for ‘cnt’ relation

In case of a UNL graph having the ‘cnt’ relation, the Punjabi word attribute of the parent node should be appended at the leftmost position in the generated output. PseudoCode 6 has been used to implement this strategy.

PseudoCode 6 (Handling of ‘cnt’ relation)

```
if (highest priority relation is ‘cnt’)
    process the active node by appending its Punjabi
        word attribute to the leftmost position in the
        generated output;
    mark the node as a special node to indicate that it
        has already processed;
end-if
```

PseudoCode 6 is inserted between lines numbered 13 and 14, and also between lines numbered 26 and 27 of PseudoCode 1 for extending it to handle the UNL graphs having the ‘cnt’ relation.

10.4 Strategy for ‘seq’ relation

The ‘seq’ relation indicates a sequence of events in the sentence. It results into insertion of ਪਹਿਲਾਂ pahilāṃ ‘before’ or ਬਾਅਦ bāad ‘after’ in the generated Punjabi sentence. When there is a sequence of events, one event generally refers to the other event. The ‘@reference’ has been used as a UNL attribute to resolve a referring event, as suggested by Nalawade (2007). This attribute may appear with the parent or child node of the ‘seq’ relation to specify the referring event.

The following strategy has been implemented for syntactic linearization of the UNL graph having the ‘seq’ relation: (1) If the parent node of the ‘seq’ relation has the ‘@reference’ attribute, then the child node of the ‘seq’ relation should be considered as the lowest priority node and it should be processed after its parent node. (2) If the child node of the ‘seq’ relation has an ‘@reference’ attribute, then the child node of ‘seq’ relation should be considered as the highest priority node and it should be traversed first.

To implement this strategy, for the ‘seq’ relation, the priority of the relation ‘seq’ has been set to the minimum out of all the relations in the priority matrix ***M***.

PseudoCodes 7 and 8 have been implemented for the syntactic linearization of the ‘seq’ relation.

PseudoCode 7 (Implementation of step 1 of the strategy for ‘seq’ relation)

```
if (highest priority relation is ‘seq’ and the active node
    has an ‘@reference’ attribute)
    process the active node by appending its Punjabi
        word attribute into the generated output;
    mark the node as a special node to indicate that it
        has already processed;
end-if
```

PseudoCode 8 (Implementation of step 2 of the strategy of the ‘seq’ relation)

```
if (active node has a ‘seq’ relation as one of its child
    relation and it does not have the ‘@reference’
    attribute)
    set ‘seq’ as a highest priority relation;
end-if
```

PseudoCode 7 is inserted between lines numbered 13 and 14, and also between lines numbered 26 and 27 of PseudoCode 1. PseudoCode 8 is inserted between lines numbered 12 and 13, and also between lines numbered 25 and 26 of PseudoCode 1. These insertions allow PseudoCode 1 to handle UNL graphs with the ‘seq’ relation.

All the algorithms and PseudoCodes discussed above have been implemented in Java to develop the proposed Punjabi DeConverter. This DeConverter forms a part of Web interface developed for online generation of Punjabi language from a given UNL expression.

11 Results and discussions

Evaluation of the proposed system has been performed by using a set of 1000 Punjabi sentences with their corresponding set of UNL expressions. For this purpose, the Spanish UNL Language Server (http://www.unl.fi.upm.es/english/index.htm) and agricultural domain threads developed by IIT, Bombay, India, are considered as gold-standard systems. The Spanish Language Server contains English sentences with their corresponding UNL expressions generated by the system. These English sentences were translated manually into equivalent Punjabi sentences for their comparison with the generated output of the Punjabi DeConverter. The agricultural domain threads developed by IIT, Bombay have Hindi language sentences with their equivalent UNL expressions. Again, these Hindi sentences were manually translated into Punjabi language for comparison with the generated output of the Punjabi DeConverter.

The UNL expressions given at these gold-standards are input to the Punjabi DeConverter for their DeConversion to Punjabi language. The output of the Punjabi DeConverter is compared with the corresponding manually translated Punjabi sentences from English and Hindi sentences given at the gold-standards. Subjective tests like adequacy and fluency tests have been performed on the proposed system. The bilingual evaluation understudy (BLEU) score has also been calculated to evaluate the quality of output. Some of the Punjabi sentences generated by the proposed Punjabi DeConverter with their corresponding input UNL expressions are given in Table 2.

The proposed system has been evaluated by 10 evaluators having a good knowledge of Punjabi and a basic knowledge of the UNL system. As suggested in LDC (2005) and Singh *et al.* (2007), the evaluators were asked to provide their intuitive reactions to the output and to work as quickly as comfortable. First of all, fluency judgments have been taken. The evaluators did not have any clue about the original source sentence for these judgments. After fluency judgments, the evaluators were asked to look at the original source sentences for adequacy judgments.

The fluency score of the proposed system is 3.61 (on a 4-point scale). The response by evaluators is further analyzed and the following are some important findings from the fluency test:

1. 73.50% sentences obtained a score 4; i.e., these are perfect having no grammatical mistake.

2. 15.50% sentences obtained a score 3; i.e., these are fair and easy to understand.

3. 9.50% sentences obtained a score 2; i.e., these are acceptable and are understandable with effort.

4. 1.50% sentences obtained a score 1; i.e., these are hard to understand.

Here, it is worth mentioning that the proposed system generated 89.0% grammatically correct sentences. These sentences are those that have a score of 3 or above.

The adequacy score of the proposed system is 3.70 (on a 4-point scale). Response by evaluators has again been further analyzed and the following are some important findings for the adequacy test.

1. 79.2% sentences obtained a score 4; i.e., there is no loss of meaning during their translation.

2. 12.8% sentences obtained a score 3; i.e., most of the meaning of these source sentences is conveyed in the translated sentence.

3. 7.0% sentences obtained a score 2; i.e., some of the meaning of these source sentences is conveyed in the translated sentence.

4. 1.0% sentences obtained a score 1; i.e., hardly any meaning of these source sentences is conveyed in the translated sentence.

Here, it is worth mentioning that the proposed system generated 92% sentences that are faithful to the original sentences. These sentences are those that have a score of 3 or above.

We have also observed that there is a strong correlation between fluency and adequacy scores. This relation between adequacy and fluency is explored further in Fig. 14. It shows the distribution of adequacy scores for various values of fluency.

Quality of the proposed system is also evaluated on the basis of the BLEU score. The proposed system is able to achieve a BLEU score of 0.72.

12 Conclusions

Syntactic linearization is an important issue in natural language generation systems. Quality of output is largely influenced by this phase of DeConverter. Matrix-based priority of relation has been used to define syntactic linearization of a UNL graph in this paper. The algorithms and pseudocodes have been implemented in Java for syntactic linearization of a simple UNL graph, syntactic linearization of a UNL graph with a scope node and to handle the untraversed parent, multiple parents, and special cases in syntactic linearization. The proposed system has been tested for 1000 UNL expressions. This system achieved a fluency score of 3.61 on a 4-point scale, an adequacy score of 3.70 on a 4-point scale, and a BLEU score of 0.72. A major outcome of this research work is the development of the Punjabi DeConverter. The proposed system can convert a UNL expression to corresponding Punjabi language text. A Web interface has also been designed for online DeConversion of the UNL expression to the corresponding Punjabi sentence. It shall enable Punjabi readers to read the

Table 2 **Punjabi sentences generated by the DeConverter with their corresponding input UNL expressions**

Sentence No.	Input UNL expression	Punjabi sentences generated by the DeConverter

1	{unl}
	mod(competition(icl>event), fair)
	obj(win(icl>do).@present.@sg.@male.@entry, prize)
	scn(win(icl>do).@present.@sg.@male.@entry, competition(icl>event))
	agt(win(icl>do).@present.@sg.@male.@entry, I(icl>person))
	{/unl}
		ਮੈਂ ਸਾਫ਼ ਸੁਥਰੇ ਮੁਕਾਬਲੇ ਵਿਚ ਇਨਾਮ ਜਿੱਤਿਆ ਹੈ ।
		maiṃ sāpha suthrē mukāblē vic inām jittiā hai.
		I have won the prize in fair competition.

2	{unl}
	obj(translate(icl>do).@present.@sg.@female.@entry, book(icl>publication))
	gol(translate(icl>do).@present.@sg.@female.@entry, punjabi(icl>language))
	src(translate(icl>do).@present.@sg.@female.@entry, English(icl>language))
	agt(translate(icl>do).@present.@sg.@female.@entry, boy(icl>young person))
	{/unl}
		ਮੁੰਡੇ ਨੇ ਅੰਗਰੇਜ਼ੀ ਤੋਂ ਪੰਜਾਬੀ ਵਿਚ ਕਿਤਾਬ ਅਨੁਵਾਦ ਕੀਤੀ ਹੈ ।
		muṇḍē nē aṅgrēzī tōṃ pañjābī vic kitāb anuvād kītī hai.
		The boy has translated the book from English to Punjabi.

3	{unl}
	aoj(boy(icl>young person), small)
	obj(eat(obj>food).@present.@sg.@male.@entry, mango)
	ins(eat(obj>food).@present.@sg.@male.@entry, spoon)
	agt(eat(obj>food).@present.@sg.@male.@entry, boy(icl>young person))
	{/unl}
		ਛੋਟੇ ਮੁੰਡੇ ਨੇ ਚੱਮਚ ਨਾਲ ਅੰਬ ਖਾਧਾ ਹੈ ।
		chōṭē muṇḍē nē cammac nāl amb khādhā hai.
		The little boy has eaten the mango with spoon.

4	{unl}
	tmt(monday, friday)
	tmf(work(icl>do).@custom.@present.@sg.@male.@entry, monday)
	agt(work(icl>do).@custom.@present.@sg.@male.@entry, he(icl>male person))
	{/unl}
		ਉਹ ਸੋਮਵਾਰ ਤੋਂ ਸ਼ੁੱਕਰਵਾਰ ਤੱਕ ਕੰਮ ਕਰਦਾ ਹੈ ।
		uh sōmvār tōṃ shukkarvār takk kamm karadā hai.
		He works from Monday to Friday.

5	{unl}
	via(go(icl>do).@custom.@present.@sg.@male.@entry, NewYork(icl>place))
	plt(go(icl>do).@custom.@present.@sg.@male.@entry, Chicago(icl>place))
	agt(go(icl>do).@custom.@present.@sg.@male.@entry, he(icl>male person))
	{/unl}
		ਉਹ ਸ਼ਿਕਾਗੋ ਨਿਊਯਾਰਕ ਵੱਲੋਂ ਜਾਂਦਾ ਹੈ ।
		uh shikāgō niūyārak vallōṃ jāndā hai.
		He goes to Chicago via New York.

6	{unl}
	plt(go(icl>do).@not.@future.@sg.@male.@entry, school)
	rsn(go(icl>do).@not.@future.@sg.@male.@entry, illness(icl>thing))
	agt(go(icl>do).@not.@future.@sg.@male.@entry, I(icl>person))
	{/unl}
		ਮੈਂ ਬਿਮਾਰੀ ਕਾਰਨ ਸਕੂਲ ਨਹੀਂ ਜਾਵਾਂਗਾ ।
		maiṃ bimārī kāran sakūl nahīṃ jāvāṅgā.
		I will not go to school because of illness.

7	{unl}
	aoj:01(green(aoj>thing).@present.@sg.@male.@entry, light(icl>thing))
	agt(go(icl>do).@present.@ability.@sg.@male.@entry, you(icl>person))
	con(go(icl>do).@present.@ability.@sg.@male.@entry, :01)
	{/unl}
		ਜੇਕਰ ਬੱਤੀ ਹਰੀ ਹੈ ਤੇ ਤੂੰ ਜਾ ਸਕਦਾ ਹੈਂ ।
		jēkar battī harī hai tē tūṃ jā sakdā haiṃ.
		If the light is green then you can go.

8	{unl}
	obj(think.@custom.@present.@sg.@female.@entry, John(icl>person))
	man(think.@custom.@present.@sg.@female.@entry, often)
	agt(think.@custom.@present.@sg.@female.@entry, she(icl>female person))
	{/unl}
		ਉਹ ਅਕਸਰ ਜਾਨ ਬਾਰੇ ਸੋਚਦੀ ਹੈ ।
		uh akasar jān bārē sōcdī hai.
		She often thinks about John.

9	{unl}
	frm(man(icl>person), Japan(icl>thing))
	obj(meet(icl>do).@present.@past.@sg.@male.@entry, man(icl>person))
	agt(meet(icl>do).@present.@past.@sg.@male.@entry, I(icl>person))
	{/unl}
		ਮੈਂ ਜਾਪਾਨ ਦੇ ਆਦਮੀ ਨੂੰ ਮਿਲਿਆ ਹਾਂ ।
		maiṃ jāpān dē ādmī nūṃ miliā hāṃ.
		I have met a man from Japan.

10	{unl}
	obj(change(icl>occur).@past.@entry, paper(icl>thing).@def)
	src(change(icl>occur).@past.@entry, red(aoj>thing))
	gol(change(icl>occur).@past.@entry, blue(aoj>thing))
	{/unl}
		ਕਾਗਜ ਲਾਲ ਤੋਂ ਨੀਲੇ ਵਿਚ ਬਦਲ ਗਿਆ ਸੀ ।
		kāgaj lāl tōṃ nīlē vic badal giā sī.
		The paper had changed from red to blue.

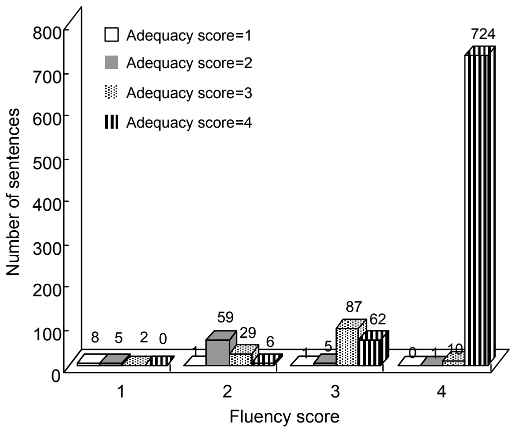

Fig. 14 Distribution of adequacy scores for various fluency scores

sentences in their local language that are originally written in different languages having their equivalent UNL expressions present on the Web. This system will also provide an opportunity to researchers working on MT to explore UNL further as an Interlingua.

References

Bhattacharyya, P., 2001. Multilingual Information Processing Through Universal Networking Language. Indo UK Workshop on Language Engineering for South Asian Languages, p.1-10.

Blanc, É., 2005. About and around the French EnConverter and the French DeConverter. *Res. Comput. Sci.*, **12**:157-166.

Boguslavsky, I., Frid, N., Iomdin, L., Kreidlin, L., Sagalova, I., Sizov, V., 2000. Creating a Universal Networking Language Module within an Advanced NLP System. 18th Int. Conf. on Computational Linguistics, p.83-89.

Boguslavsky, I., Cardeñosa, J., Gallardo, C., Iraola, L., 2005. The UNL initiative: an overview. *LNCS*, **3406**:377-387.

Daoud, M., 2005. Arabic generation in the framework of the Universal Networking Language. *Res. Comput. Sci.*, **12**:195-209.

Dave, S., Parikh, J., Bhattacharyya, P., 2001. Interlingua based English Hindi machine translation and language divergence. *J. Mach. Transl.*, **16**(4):251-304.

Dey, K., Bhattacharyya, P., 2005. Universal Networking Language based analysis and generation of Bengali case structure constructs. *Res. Comput. Sci.*, **12**:215-229.

Dhanabalan, T., Geetha, V., 2003. UNL DeConverter for Tamil. Int. Conf. on the Convergence of Knowledge, Culture, Language and Information Technologies, p.1-6.

Hrushikesh, B., 2002. Towards Marathi Sentence Generation from Universal Networking Language. MT Thesis, Indian Institute of Technology, Bombay, Mumbai.

Keshari, B., Bista, K., 2005. UNL Nepali DeConverter. 3rd Int. Conf. on CALIBER, p.70-76.

Lewis, M.P., 2009. Ethnologue: Languages of the World (16th Ed.). SIL International, Dallas.

Linguistic Data Consortium (LDC), 2005. Linguistic Data Annotation Specification: Assessment of Adequacy and Fluency in Translations. Revision 1.5, Technical Report.

Martins, T., Rino, M., Osvaldo, N., Hasegawa, R., Nunes, V., 1997. Specification of the UNL-Portuguese EnConverter-DeConverter Prototype. Relatórios Técnicos do ICMC-USP, p.1-10.

Nalawade, A., 2007. Natural Language Generation from Universal Networking Language. MT Thesis, Indian Institute of Technology, Bombay, Mumbai.

Pelizzoni, J., Nunes, M., 2005. Flexibility, configurability and optimality in UNL DeConversion via multiparadigm programming. *Res. Comput. Sci.*, **12**:175-194.

Raman, S., Alwar, N., 1990. An AI-based approach to machine translation in Indian languages. *Commun. ACM*, **33**(5): 521-527.

Shi, X., Chen, Y., 2005. A UNL DeConverter for Chinese Universal Network Language. *Res. Comput. Sci.*, **12**: 167-174.

Singh, S., Dalal, M., Vachhani, V., Bhattacharyya, P., Damani, O.P., 2007. Hindi Generation from Interlingua. 17th MT Summit, Copenhagen, Denmark, p.1-8.

Sinha, R., 2005. Hindi Generation: Syntax Planning and Case Marking. Mini Project Report, Indian Institute of Technology, Bombay, Mumbai.

Uchida, H., 2005. Universal Networking Language (UNL): Specifications Version 2005, UNDL Foundation.

Vachhani, V., 2006. UNL to Hindi DeConverter. BE Thesis, Dharamsinh Desai Institute of Technology, Nadiad.

Vora, A., 2002. Generation of Hindi sentences from Universal Networking Language. BE Thesis, Dharamsinh Desai Institute of Technology, Nadiad.