



**Addis Ababa University**  
**College of Natural and Computational Sciences**  
**Department of Computer Science**  
**Natural Language Processing (CoSc 6405)**

**Project on**  
**Statistical Machine Translator from Amharic to English**

To: Yaregal Assabie (PhD)  
By: Rabra Hierpa  
GES/6846/12  
Aug 25, 2020

## Table of Contents

1. Introduction .....	1
2. Problem Statements .....	1
3. Specific Objectives .....	2
3.1. Specific Objectives .....	2
4. Method .....	2
4.1. Tool installation .....	3
4.2. Corpus Preparation .....	4
4.3. Language Model Training .....	6
4.4. Training the Translation System .....	7
5. Implementation.....	8
5.1. Challenges to the development of the system .....	9
5.2. Performance of the system .....	9
5.3. Limitation of the system.....	9
6. Future Work .....	9
References .....	10

# 1. Introduction

Natural language processing (NLP) is one of the most promising fields in computer science research which have the potential to enable humans to communicate or command computers in a natural way as humans do. One aspect of NLP is Machine Translation. Machine translation (MT) is sub-field of computational linguistics that investigates the use of computer software to translate text or speech from one language to another. At its basic level, MT performs simple substitution of words in one natural language from words in another.[1]

There are different translation models available but the most common ones are statistical machine translation, rule-based machine translation and example-based machine translation. In this project we are interested on statistical machine translation. Statistical machine translation is a machine translation paradigm where translations are generated on the basis of statistical models.[1] These statistical model parameters are derived from the analysis of bilingual text corpora. In this project, proposed Amharic to English statistical machine translation has been presented.

## 2. Problem Statements

Some of the difficulties in performing automatic translation is that there are several structural and stylistic differences among languages. The issues are

**Word order** – Every language follows a certain type of word ordering. Some of the classifications can be done by naming the typical order of subject(S), verb(V) and object(O). Some languages follow an SVO word order (such is the case with the English Language) and some follow SOV (such is the case with the Amharic Language). Hence, word to word translation is difficult to perform in such cases.

**Word sense** – The same word may give different meaning when being translated to another language. Word diction or selection is critical to translate the meaning of a text with its specific context.

**Pronoun resolution** – The problem of not resolving the pronominal references is important for machine translation. Unresolved reference can lead to incorrect translation.

**Ambiguity** - Ambiguity can be referred as the ability of having more than one meaning or being understood in more than one way. Natural languages are ambiguous, so computers are not able to understand language the way people do. [3]

The need to develop a machine translation system for Amharic language is essential because most of the historical documents in Ethiopia are not available in the English format. This project tries to develop a statistical machine translation system by taking a parallel corpus from the Federal Civil Servants proclamation which is available from the Federal Negarit Gazette.

### 3. Specific Objectives

The general objective of this project is to develop a Statistical Machine Translation system that translates Amharic language to English Language.

#### 3.1. Specific Objectives

The specific objectives are

- Prepare Amharic and English parallel bilingual corpora
- Align Amharic word with English
- Generate a monolingual language model
- Decode the language model
- Train the translation model based on the prepared corpus
- Test the translation system

### 4. Method

For developing the Amharic to English statistical machine translation, different tools were used. The tools are

- **Moses** for decoding
- **GIZA++** for word-aligning our parallel corpus
- **IRSTML** for language model estimation.
- **Sublime Text** to make the corpus in machine editable way.
- **xUbuntu 19.04** used for compiling the above tools and developing the translation system.

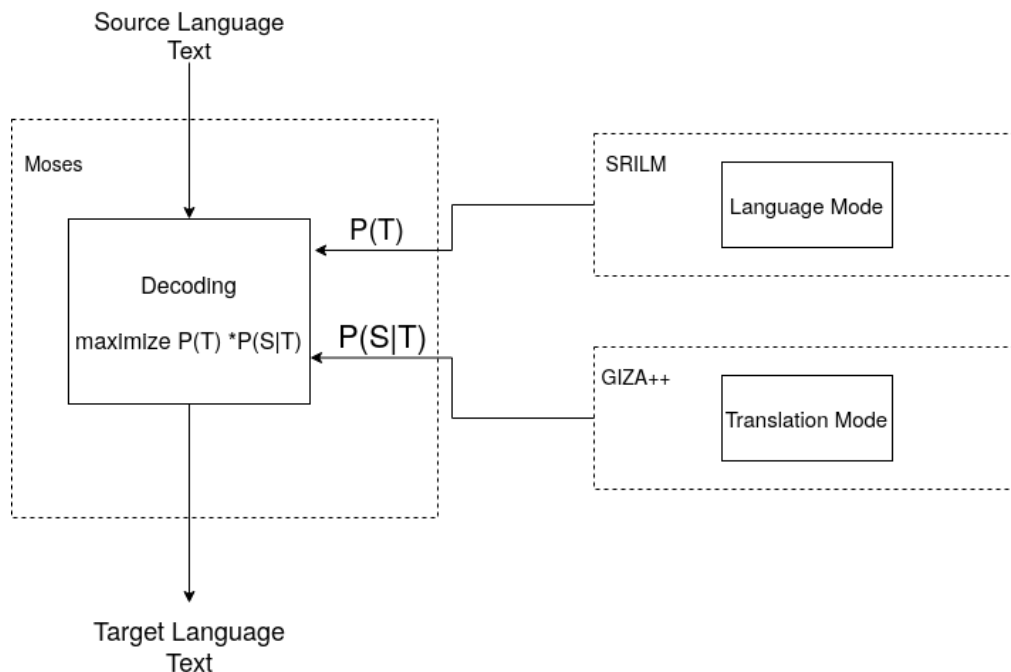


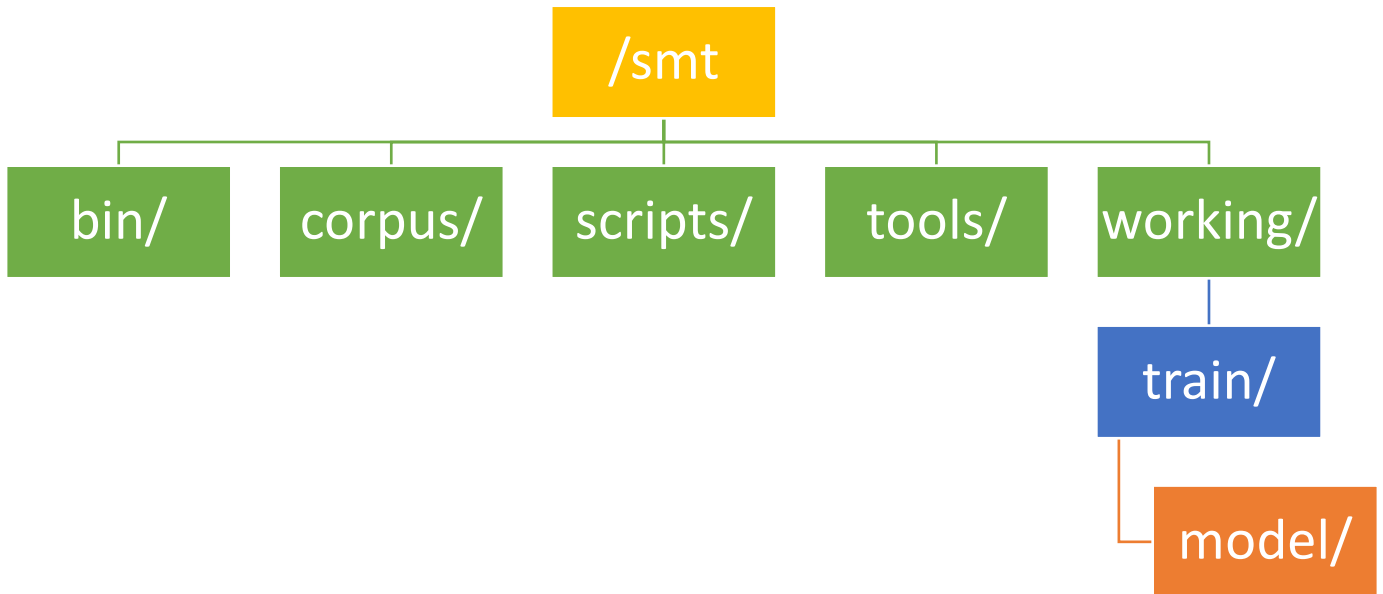
Figure 1 Translation System Model

Steps used for developing the Statistical machine translation system are as follows

#### 4.1. Tool installation

In order to develop an SMT we need to have the necessary tools installed on our machine. As mentioned above the xUbuntu18.04 OS is used to install the tools, develop the system and finally test the prototype. A detailed instruction to install all the necessary tools mentioned above is found at <https://bily.ly/installing-moses-for-smt>

In order to have an ordered directory structure, the following structure was used.



*Figure 2 Directory structure of the project*

- **smt/** - contains all the tools that are required to build the system.
- **bin/, scripts/, tools/** - contain scripts that are used to tokenize, Ture case and clean the corpus data.
- **corpus/-** contains the initial data set for all the tokenization, true casing, cleaning, training and testing.
- **working/-** the actual directory where the trained data and the initial mooses.ini file is generated.

This organization of the directory structure makes it easy to follow along executing the commands in the following sub sections.

## 4.2. Corpus Preparation

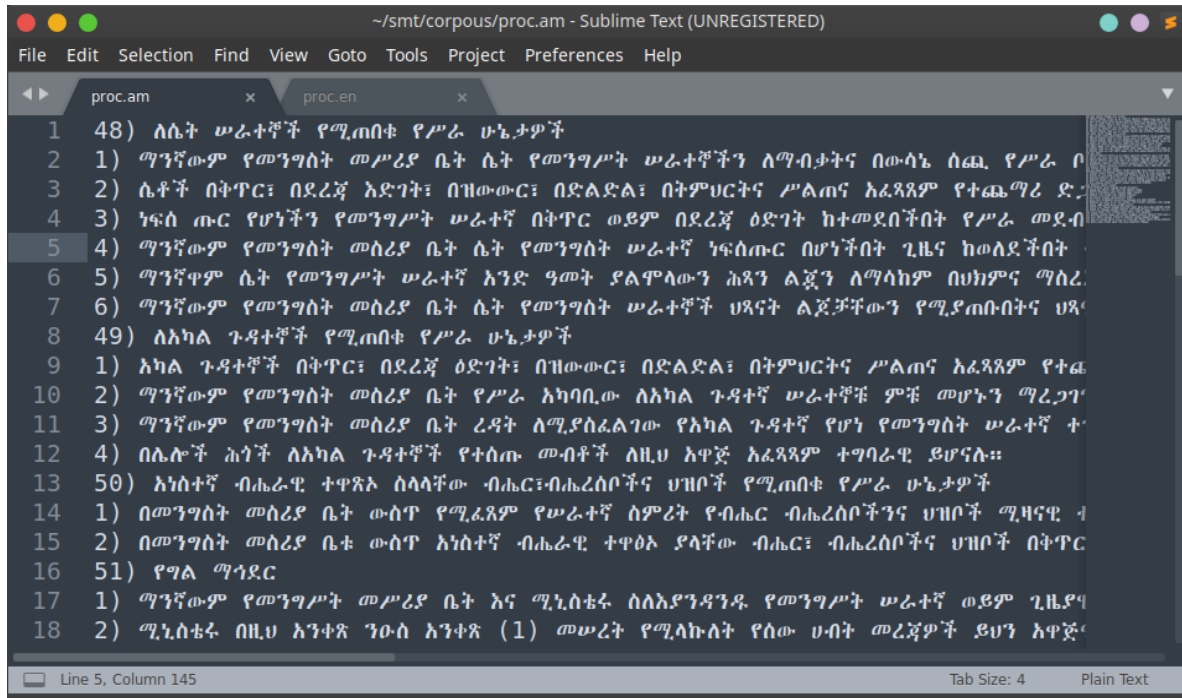


Figure 3 Amharic Corpus

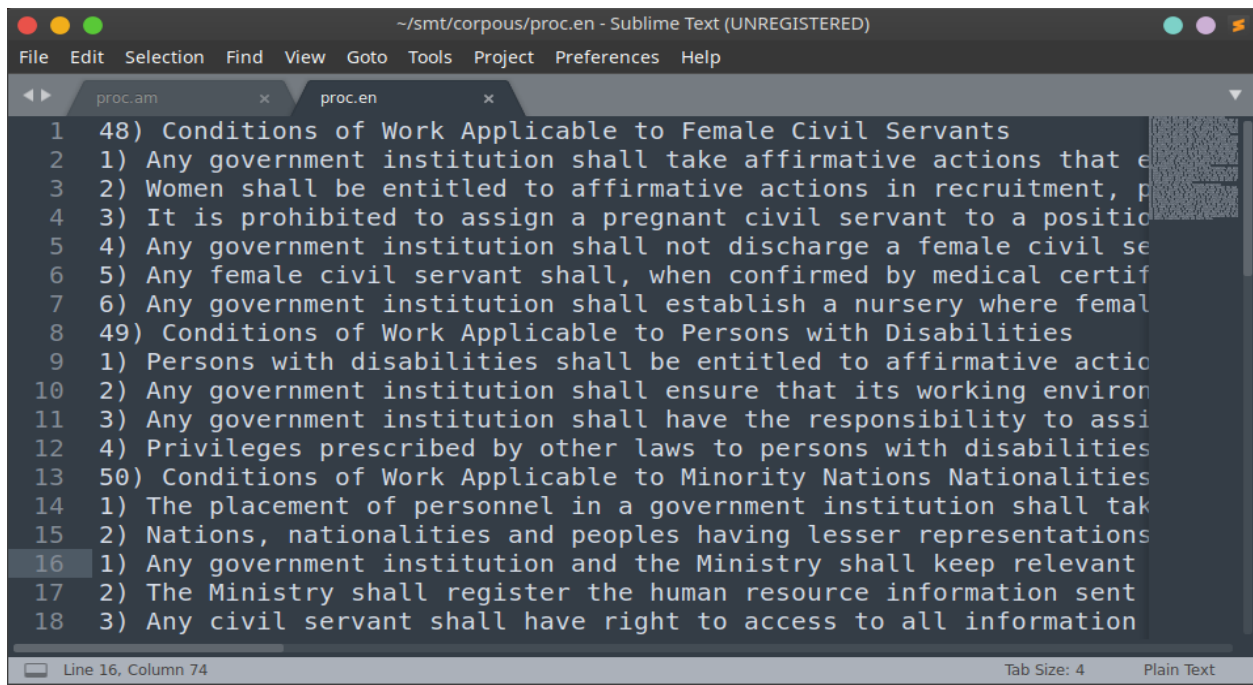


Figure 4 English Corpus

The corpus data was prepared in a format that needs to be applicable for translation. The initial corpus data are named proc.ah for Amharic and proc.en for English. To prepare the data for training the translation system the following steps need to be performed [2]:

- **Tokenization:** This is the step where the corpus is split from a sentence or paragraph in to smaller individual units such as words or terms. Each of these smaller units is called token. To tokenize the corpus the following commands was issued.

```
## tokenize the English corpus
$ ./mosesdecoder/scripts/tokenizer/tokenizer.perl -l en
  < ./corpus/proc.en # take proc.en corpus as a file
  > ./corpus/proc.tok.en # write the tokenized output to proc.tok.en
file
## tokenize the Amharic corpus
$ ./mosesdecoder/scripts/tokenizer/tokenizer.perl -l ah
  < ./corpus/proc.ah # take proc.en corpus as a file
  > ./corpus/proc.tok.ah # write the tokenized output to proc.tok.en
file
```

- **Truecasing:** The first word of each sentence is converted to the most probable casing. This can help reduce data sparsely. In order to do truecasing, we need to have a model which can be generated by training the system from a tokenized corpus. To do truecasing for the corpus the following commands were issued.

```
## training the caser for English corpus
sudo ./mosesdecoder/scripts/recaser/train-truecaser.perl
  --model ./corpus/truecase-model.en # output file for the english
case model
  --corpus ./corpus/proc.tok.en # input for the case learner
```

After successfully training the case model for the English language, the tokenized corpus was cased by issuing the following commands

```
## casing the English corpus
sudo ./mosesdecoder/scripts/recaser/truecase.perl
  --model ./corpus/truecase-model.en # model config file
  < ./corpus/proc.tok.en # input for the caser
  > ./corpus/proc.true.en # output file after casing the corpus
```

Since Amharic language doesn't have upper and lower cases this step for the Amharic language is skipped.

- **Cleaning:** for this phase a small script is used to clean the corpus by removing empty lines and redundant space characters. The sentence length is also limited to 80. We can do that by issuing a single command.

```
## cleaning both corpus
sudo ./mosesdecoder/scripts/training/clean-corpus-n.perl
    ./corpus/proc.true ah en # take both cased files as an input
    ./corpus/proc.clean 1 80 # limit sentence length to 80
```

### 4.3. Language Model Training

After successfully passing the tokenization, turecaseing and cleaning phase, we build the language model (LM). The language model is used to ensure fluent output, so it is built with the target language (i.e. English in this case). An appropriate 3-gram language model will be built by the following command.

```
sudo ./mosesdecoder/bin/lmplz -o 3
    < ./corpus/proc.true.en # turecased corpus input file
    > ./corpus/proc.arpa.en # the target language model(i.e English in
our case)
    # Then we should binarize (for faster loading) the *.arpa.en file
using KenLM:
sudo ./mosesdecoder/bin/build_binary
    ./corpus/proc.arpa.en # the input file to be binarize
    ./corpus/proc.blm.en # the binary file of the language model
```

The language model was tested by the following query:

```
echo "የመንግሥት ሠራተኛው እንዲያውቀው ያልተደረገ ወይም ያልተገለጸለትን የጽሁፍ ማስረጃ
በግል ማህደሩ ውስጥ ማስቀመጥ ክልክል ነው።" | # redirect input to next command
./bin/query # the echo as an input to the query script file
./corpus/proc.blm.en # the binary language model file
```



```

rz@xubuntu2: ~/smt
rz@xubuntu2: ~/smt 90x17
~/smt
→ echo "የመንግሥት ሀይተኛው አንዲያውቀው ያልተደረገ ወይም ያልተገለጸለትን የጽሁፍ ማስረጃ በግል ማግኘት ወስኖ ማስቀመጥ ክልክል ነገር፡"
| ./bin/query ./corpous/proc.blm.en
የመንግሥት=0 1 -3.328314 ሀይተኛው=0 1 -2.9134054 አንዲያውቀው=0 1 -2.9134054 ያልተደረገ=0 1 -2.9134
054 ወይም=0 1 -2.9134054 ያልተገለጸለትን=0 1 -2.9134054 የጽሁፍ=0 1 -2.9134054 ማስ
ረጃ=0 1 -2.9134054 በግል=0 1 -2.9134054 ማግኘት=0 1 -2.9134054 ወስኖ=0 1 -2.9134054
ማስቀመጥ=0 1 -2.9134054 ክልክል=0 1 -2.9134054 ነገር፡=0 1 -2.9134054 </s>=2 1 -2.066671
Total: -43.26926 OOVs: 14
Perplexity including OOVs: 766.685695474101
Perplexity excluding OOVs: 116.5939193201424
OOVs: 14
Tokens: 15
Name:query VmPeak:29508 kB VmRSS:5624 kB RSSMax:5672 kB user:0.008873 sys:0.0088
73 CPU:0.017746 real:0.0043122
~/smt
→ |

```

Figure 5 Language Model Test

## 4.4. Training the Translation System

In this final phase, we train the translation mode. To do this, we run word alignment (using GIZA++), phrase extraction and scoring, create lexicalized reordering tables and create our Moses configuration file, all with a single command.

```

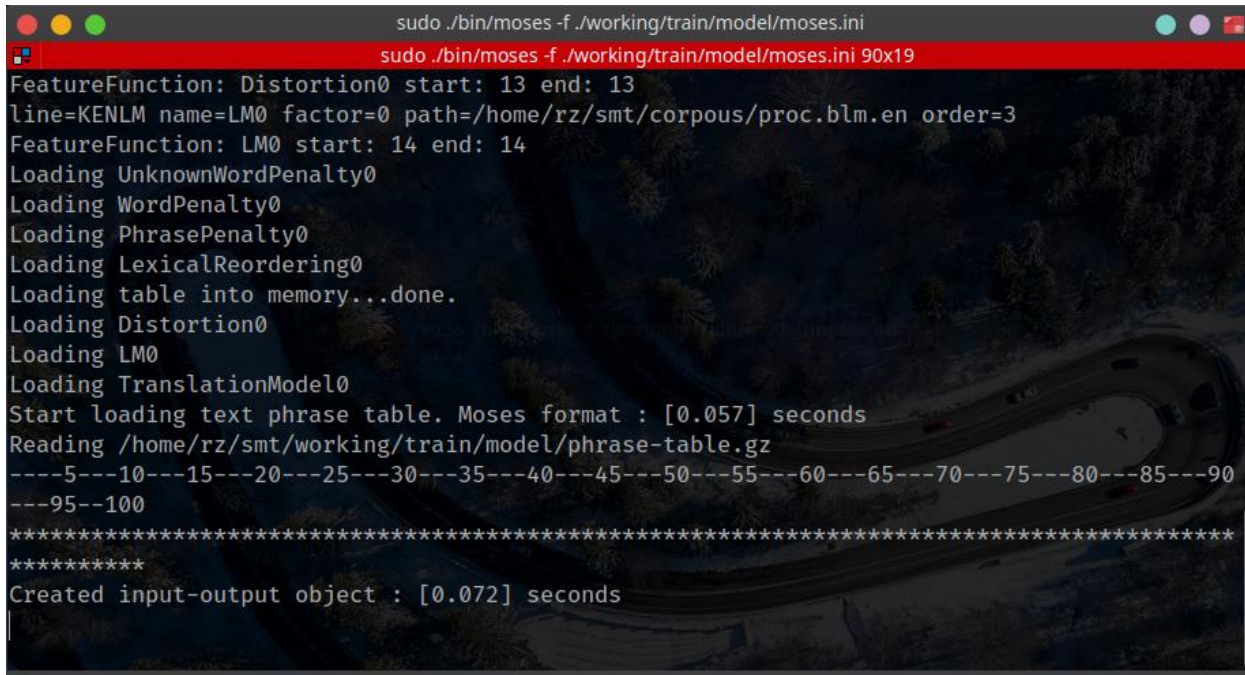
## the following command is executed in the working/ directory
nohup nice # runs the command in the background
../scripts/training/train-model.perl # the training script
-root-dir train # output dir
-corpus ../corpous/proc.clean -f ah -e en # the cleaned corpus files
-alignment grow-diag-final-and # word alignment
-reordering msd-bidirectional-fe # lexicalized reordering
-lm 0:3:$HOME/smt/corpous/proc.blm.en:8 # binary file of the
language model and the path here must be absolute
-external-bin-dir ../tools # dir where GIAZ++,mkcsl and sn2cooc.out
are located
>& training.out & # write the status to training.out file

```

## 5. Implementation

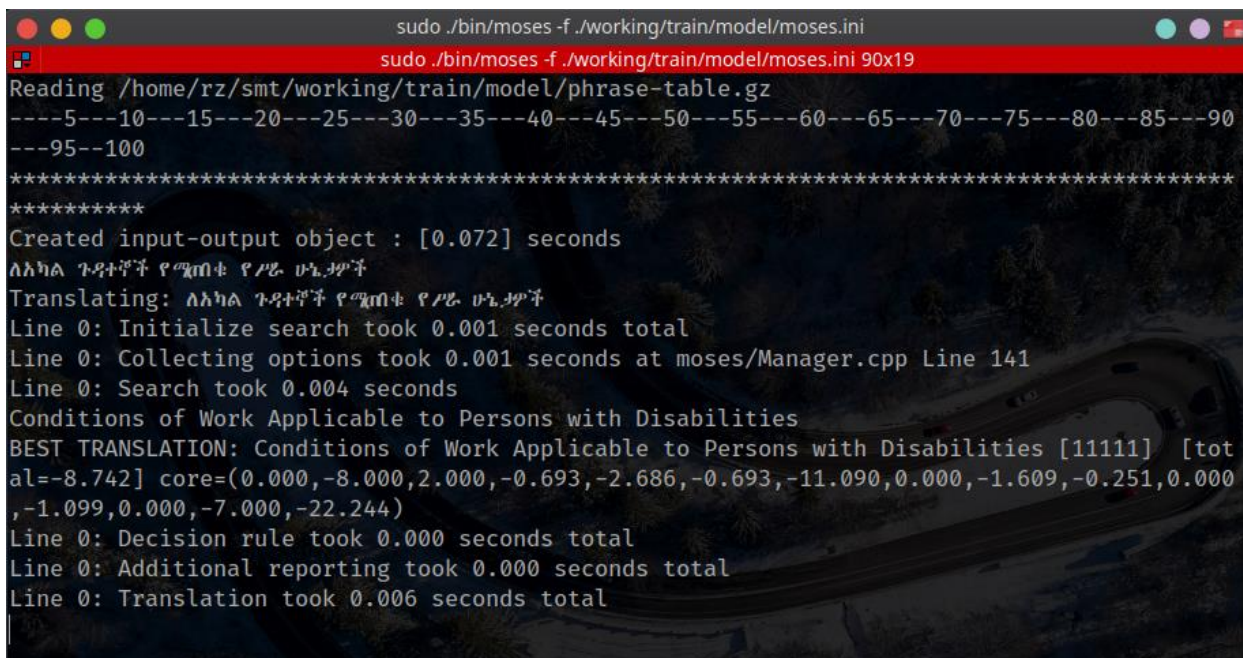
In this phase, the prototype of the system is queried to translate a given text in Amharic to English language. This can be done by a single command

```
sudo ./bin/moses -f ./working/train/model/moses.ini
```



```
sudo ./bin/moses -f ./working/train/model/moses.ini
sudo ./bin/moses -f ./working/train/model/moses.ini 90x19
FeatureFunction: Distortion0 start: 13 end: 13
line=KENLM name=LM0 factor=0 path=/home/rz/smt/corpus/proc.blm.en order=3
FeatureFunction: LM0 start: 14 end: 14
Loading UnknownWordPenalty0
Loading WordPenalty0
Loading PhrasePenalty0
Loading LexicalReordering0
Loading table into memory...done.
Loading Distortion0
Loading LM0
Loading TranslationModel0
Start loading text phrase table. Moses format : [0.057] seconds
Reading /home/rz/smt/working/train/model/phrase-table.gz
----5---10---15---20---25---30---35---40---45---50---55---60---65---70---75---80---85---90
---95---100
*****
*****
Created input-output object : [0.072] seconds
```

Figure 7 Running Moses



```
sudo ./bin/moses -f ./working/train/model/moses.ini
sudo ./bin/moses -f ./working/train/model/moses.ini 90x19
Reading /home/rz/smt/working/train/model/phrase-table.gz
----5---10---15---20---25---30---35---40---45---50---55---60---65---70---75---80---85---90
---95---100
*****
*****
Created input-output object : [0.072] seconds
ለአካል ጉዳተኞች የሚጠበቁ የሥራ ሁኔታዎች
Translating: ለአካል ጉዳተኞች የሚጠበቁ የሥራ ሁኔታዎች
Line 0: Initialize search took 0.001 seconds total
Line 0: Collecting options took 0.001 seconds at moses/Manager.cpp Line 141
Line 0: Search took 0.004 seconds
Conditions of Work Applicable to Persons with Disabilities
BEST TRANSLATION: Conditions of Work Applicable to Persons with Disabilities [11111] [tot
al=-8.742] core=(0.000,-8.000,2.000,-0.693,-2.686,-0.693,-11.090,0.000,-1.609,-0.251,0.000
,-1.099,0.000,-7.000,-22.244)
Line 0: Decision rule took 0.000 seconds total
Line 0: Additional reporting took 0.000 seconds total
Line 0: Translation took 0.006 seconds total
```

Figure 6 Sample Translation

### 5.1. Challenges

The challenges encountered during the development of the system are listed as follows:

- Lack of clear resources to understand statistical machine translation tools.
- Lack of well encoded parallel corpus. Some of the published Federal Negarit Gazette digital copies had no proper encoding of the Amharic characters.
- Compiling and building the Moses tools takes a lot of time.

### 5.2. Performance of the system

The system has the following performance characteristics:

- The system responds to user queries within a fraction of seconds even with long sentences.
- If the Amharic sentence is in the corpus the system translates correctly.

### 5.3. Limitation of the system

The system has the following limitations:

- The system has low accuracy when translating long sentences.
- The translation of non-existing word is partially unsatisfactory; It only translates some of the words and replies back the words it does not recognize.
- When translating long sentences the last word is replaced by the Amharic word and some space characters are replaced by “|UKN|”.

## 6. Future Work

Based on this project, we can see that it is possible to build a statistical machine translator for Amharic to English language translation. The limitation of the system can be improved by providing huge volume of parallel bilingual corpus. It is also possible to integrate this system to a user-friendly application by integrating a Graphical User Interface with tools like Python or Java.

## References

- [1]. Philipp Koehn, "Europarl: A Parallel Corpus for Statistical Machine Translation", MT Summit, 2005
- [2]. Achraf Othman, Mohamed Jemin, "[Designing High Accuracy Statistical Machine Translation for Sign Language Using Parallel Corpus: Case Study English and American Sign Language](#)", Journal of Information Technology Research, Volume 12, Issue 2, April-June 2019
- [3]. Anjali M.K, Babu Anto P, "Ambiguities in Natural Language Processing", International Journal of Innovative Research in Computer and Communication Engineering, Vol2, Issue 5, Oct 2012, India