Visvesvaraya National Institute of Technology,

Nagpur

NATURAL LANGUAGE TO STRUCTURED ENGLISH TRANSLATION

A COMPARATIVE STUDY OF MACHINE TRANSLATION APPROACHES

BY

RAKSHITA RAGURAMAN

HARSHADA KUMBHARE

DEWANG PALAV

ABHIJEET KRISHNAN

A thesis submitted in partial fulfilment for the degree of

BACHELORS OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Under the Guidance of

DR. UMESH A. DESHPANDE, VNIT, NAGPUR

DR. SAGAR SUNKLE, TRDDC, PUNE

CERTIFICATE

This is to certify that “Natural Language to Structured English Translation: A comparative study of two Machine Translation approaches" is being submitted by Rakshita Raguraman, Harshada Kumbhare, Dewang Palav and Abhijeet Krishnan in the partial fulfillment of the requirement for the completion of the course for the degree of Bachelor of Technology in Computer Science and Engineering, is a record of the students' own work carried out by them under my supervision and guidance. This work is comprehensive, complete and fit for evaluation.

Dr. Umesh A. Deshpande
Head of Department and Project Guide

ACKNOWLEDGEMENT

We begin with heartfelt gratitude towards of project guide Dr. U. A. Deshpande, Head of Department, Computer Science and Engineering, VNIT, Nagpur to presenting us with an opportunity to pursue a project which was close to our interests and establishing an efficient line of communication about the project with the industry experts. It is because of his continued support we could learn and grow during the process.

We thank Vinay Kulkarni, TRDDC, Pune, for granting us to work on a project in association with the research happening in the premier organization.

We are indebted to Dr. Sagar Sunkle, TRDDC, Pune, who has been extremely patient and motivating throughout the journey. We thank him for answering all our queries and encouraging our endeavors by allowing us to come with alternative approaches to the problem.

We sincerely thank the Department of Computer Science and Engineering, VNIT Nagpur, for granting us unrestricted access to the Department laboratory.

We are grateful towards the entire faculty of Department of Computer Science and Engineering, VNIT Nagpur, for their support. We would also like to thank our fellow students from the subsequent part of the larger project who have helped us and encouraged us through this journey.

ABSTRACT

Regulatory compliance is a vital requirement for modern-day enterprises. As a number of new regulations are imposed frequently, it is very difficult to process these and take required measures without the help of an expert. The aim is to reduce this effort by converting Natural Language to executable rules. This process first involves conversion of Natural language to a Structured English format. Further, SBVR specification needs to be developed using Structured English, which is finally converted to executable rules using Drools.

The use of Machine Learning in the process of Natural language to Structured English translation is analysed in the project. It involves reviewing existing literature on the use of ML tools in the translation of Regulatory documents to codified regulations. The two technologies evaluated using a common corpus are **Moses** (Statistical Machine Translation) and **Tensorflow** (Neural Machine Translation). The challenges involved in the use of these open source technologies are addressed and enumerated. The different parameters that govern the translation paradigms are identified and modified. The results obtained using both the tools are tabulated.

CONTENTS

1 . INTRODUCTION

Natural language to Structured English translation is a domain meant to aid rule generation from regulatory documents. This is first step of a pipeline of transformation which involves use of NL processing tools. The existing work in this field is in its initial stage with many issues and lack of standard approaches. The use of Machine Learning to translate Natural Language (NL) texts to a more compressed version termed as Structured English (SE) has been advocated in recent times. This gives rise to evaluation of existing tools and technologies in this domain.

The following is a basic example of the two ends:
NL : *Each operating company must have at least one insurer.*
SE : *It is obligatory that each operating company has at least one insurer.*

The project involves study of Statistical Machine Translation (SMT) and Neural Machine Translation (NMT) . The representative tools used are Moses and Tensorflow in the respective paradigms. The process involves deriving a corpus from EU-Rent (a document with standardised examples from NL and SE) . The challenges faced during the use of open source tools and the results obtained from the corpus are enumerated. The inferences and remarks from the results are tabulated. The major challenge in identifying the parameters that provide variations in the use of tools is addressed. Finally the future scope of the use of the tools is listed with an alternative approach suggested.

2 . OVERVIEW

This section presents the review of related papers and basic structure of NL and SE are explained.

2.1 LITERATURE REVIEW

Our project presents a comparative study on various machine translation techniques as applied to the task of translating natural language rules in legal texts into a Structured English format. We use the techniques of Statistical Machine Translation (SMT) and Neural Machine Translation (NMT). In this chapter, we present a summary of some existing work that has been done in this field.

Andreas Stolcke [14] presents a general overview on SRILM, which is a language modelling toolkit to develop language models to be used in machine translation. It begins with describing the design goals of the software and gives a brief description of its functionality. It mentions some additional features also available in the toolkit. It then describes the actual software design and concludes with future improvements. SRILM is a language model that is supported by Moses (although it is not used by default) and is necessary to run the EMS (Experiment Management System) for Incremental Training.

Kishore Papineni et al. [15] presents BLEU as a method to automatically evaluate machine translations to assess their quality. They begin by arguing for the need for such a metric, and follow by presenting some background details. They describe their method as utilising n-gram precisions, along with weighing other factors such as translation brevity. This is the metric used in Moses to evaluate translations.

Adam Lopez [16] presents a summary of current research work done in the field of SMT. He goes on to present a technical overview of SMT and the various steps involved, such as the formalisation of the problem to be solved, the various models to translate text (FST, SCFG), parameterisation using generative models and parameter estimation (log-linear models and MERT are described, which is used in Moses), and finally decoding a sentence. The author concludes with describing some metrics for evaluation and the current and future research.

Koehn et al. [17] describe an open-source SMT toolkit they have developed called Moses. They describe the motivation behind creating the toolkit. They also describe its features, such as confusion networks for translating speech-to-text corpuses, and support for factored translation models. They describe some of its performance optimisations which greatly reduce the time taken to create a model and translate a sentence.

Nguyen Bach and Sameer Badaskar [18] present a moderately technical overview of the various algorithms and techniques that have been developed for relation extraction in unstructured text. The methods they present are supervised approaches such as feature based and kernel based (tree, subsequence, dependency tree), and semi-supervised approaches such as DIPRE and Snowball. They also review a method to extract higher-order relations (all the previous methods merely extract binary relations) and discuss evaluation methods and standard datasets. The motivation to choose DIPRE as the relation classifier of choice becomes apparent in this paper, since it is described as requiring a seed example to begin classification, and then iteratively improves its classification by extracting patterns from the seed relations. This is also a weakness, since it uses hard pattern matching, which is probably why the researchers from TRRDC opted for IE in their system.

Sagar Sunkle, Deepali Kholkar and Vinay Kulkarni [19] studied the existing natural language processing and machine learning techniques used to extract the rules from legal NL texts in a semi-automated manner. They identified problems with the existing approaches such as the requirement of domain experts to identify structural arrangements in the text to aid analyses. They argue that a generic approach that uses conceptual method modeling should drive the legal rule extraction. Using fact orientation (FO) as the overall modeling method, they go about the task in three steps - a) create a domain model and dictionary using FO and relation extraction (RE) b) Active Learning to identify rules and c) Converting the Domain Model, Dictionary and Rules into a formal specification. Step (a) was accomplished using the LingPipe toolkit to find mentions of concept types and using Ollie, an open source implementation of open IE (Information Extraction). Step (b) was accomplished using the LingPipe toolkit and Step (c) was accomplished using the NORMA tool. The algorithms were tested on a KYC guidelines document issued by the RBI. The initial results are encouraging and show that we can do away with simplified paraphrasing of legal NL texts and other annotations used in existing approaches while reliably generating a domain model and a dictionary.

Sagar Sunkle, Deepali Kholkar and Vinay Kulkarni [20] expand on their earlier work described in “Comparison and Synergy Between Fact-Orientation and Relation Extraction for Domain Model Generation in Regulatory Compliance,” and describe in further detail the active learning approach taken for rule identification. The initial steps are identical to the previous paper. The reason for choosing active learning is due to it requiring very few labeled sentences to learn. The LingPipe toolkit is used to extract features and implement a classifier. Informed active learning is used, which asks the domain expert to classify rules when the classifier is least certain about a rule (uncertainty sampling) or most certain about a rule (certainty sampling). Performance comparisons are presented when a) a dictionary and domain model are used b) a feature extractor based on n-gram tokenizer is used c) both (a) and (b) are used. All three comparisons are carried out for both certainty and uncertainty sampling and the results are summarized.

Sagar Sunkle, Deepali Kholkar and Vinay Kulkarni [21] present again their approach to rule identification using fact orientation, relation extraction and active learning. They also discuss two approaches for authoring rules, namely Rule Authoring Editor and Active Semantic Parser. The first approach is easier to implement but the second approach is more promising. Also discussed is the possibility of using SMT with a negative translation memory.

2.2 STRUCTURE OF NL AND SE

The general structure of SE, according to the OMG SBVR Documentation[2] includes the following :

• '**Modality qualifiers**' indicates the verb concept that is central to the element of guidance, indicating the kind of situation that is obligatory, necessary, permitted or possible.

• '**Supporting verb concepts**' are the additional verb concepts on which the guidance is directly based. They qualify or quantify the roles in the central verb concept

Examples of SE:

It is obligatory that each driver who is authorised for a rental is qualified.

It is prohibited that the rental duration of a rental is greater than 90 rental days.

It is permitted that a rental is open only if an estimated rental price of the rental is provisionally charged to a credit card that is in the name of the renter who is responsible for the rental.

It is necessary that the rental price has exactly one rental period.

2.3 NATURAL LANGUAGE PROCESSING

Natural Language Processing aims to perform computations on natural language and speech in order to perform useful tasks. It is a vast field encompassing various problems.

The document analysed in our project is: **‘EU-Rent’**

It is an Annex which presents a fictional company with regulations for transport rentals in Natural Language to be converted into Structured English (SVBR).

Given the complexity of natural language, NLP is considered an **AI-hard or AI-complete** problem as full understanding of natural language is illusive.

Complexity of NLP

Natural language sentences are basically means humans use to capture thoughts.
Representing all thoughts in AI seems quite complex.
There are umpteen examples of ambiguities in sentences even for the human reader.

For eg: (including one of many from EURent)

1. *Alice yelled at Eve and <u>she</u> cried.*

   Here, she may represent either Alice or Eve
2. *An open rental that has a rented car <u>that is due for service</u> or is in need of repair must receive a car exchange during rental.*

   Here for the human mind the term ‘that is due for service’ applies to the car, but could apply to ‘open rental’ too for a machine.
3. *I'm glad I'm a man, and so is Lola. — Lola by Ray Davies*

   Either ‘I’ and ‘Lola’ both are happy or both ‘I’ and ‘Lola’ are men.

STATISTICAL MACHINE TRANSLATION USING MOSES

3 . STATISTICAL MACHINE TRANSLATION USING MOSES

Statistical Machine Translation is a Machine Translation model in which translations are produced using statistical models. The parameters used in this statistical model are derived by analysing monolingual and bilingual corpora. Statistical Machine Translation model can be easily fitted to any pair of languages.

MOSES :

Moses is one of the most popular open source statistical machine translation engines. It is written in C++ along with supporting scripts in various languages . It supports phrase based (explained in detail in next section) , hierarchical phrase based (no linguistic syntax) and syntax based (linguistic syntax) Machine Translation systems.

3.1 SELECTION AND INSTALLATION

Moses is the most popular open-source statistical MT tool. It has extensive documentation and provides a lot of flexibility. Hence, the selection.

System Configuration :

Platform recommended : Linux (14.04 LTS)

Ram : 8+ GB recommended

Disk size : 10+ GB

Installation of Natural Language Toolkit (NLTK) :

NLTK is required to do preprocessing on the corpus.

Note: Python3 and pip3 should be already installed.

Open Terminal and enter following commands:

```
$ sudo pip3 install -U nltk
```

To download all dataset and models:

```
$ python3
>>> import nltk
>>> nltk.download(‘all')
```

Installation of Moses :

Open terminal and do following 5 steps:

1. Install following Ubuntu packages to build Moses and its dependencies:

```
$ sudo apt-get install build-essential git-core pkg-config
automake libtool wget zlib1g-dev python-dev libbz2-dev
```

To run regression tests.

```
$ sudo apt-get install libsoap-lite-perl
```

2. Clone Moses from the repository and change current directory to mosesdecoder

```
$ git clone https://github.com/moses-smt/mosesdecoder.git
$ cd mosesdecoder
```

3. To install a recent version of Boost

```
$ make -f contrib/Makefiles/install-dependencies.gmake
```

4. To compile Moses , run

```
$ ./compile.sh
```

5. To install bjam

```
$ wget http://downloads.sourceforge.net/project/boost/
boost/1.55.0/boost_1_55_0.tar.gz
$ tar zxvf boost_1_55_0.tar.gz
$ cd boost_1_55_0/
$ ./bootstrap.sh
$ ./b2 -j4 --prefix=$PWD --libdir=$PWD/lib64 --
layout=system link=static install || echo FAILURE
```

To compile with required and minimum of features:

```
$ ./bjam -j4
```

Installation of MGIZA++ :

MGIZA++ is a system for training word alignment systems. It is multi-threaded. Run following commands to install it :

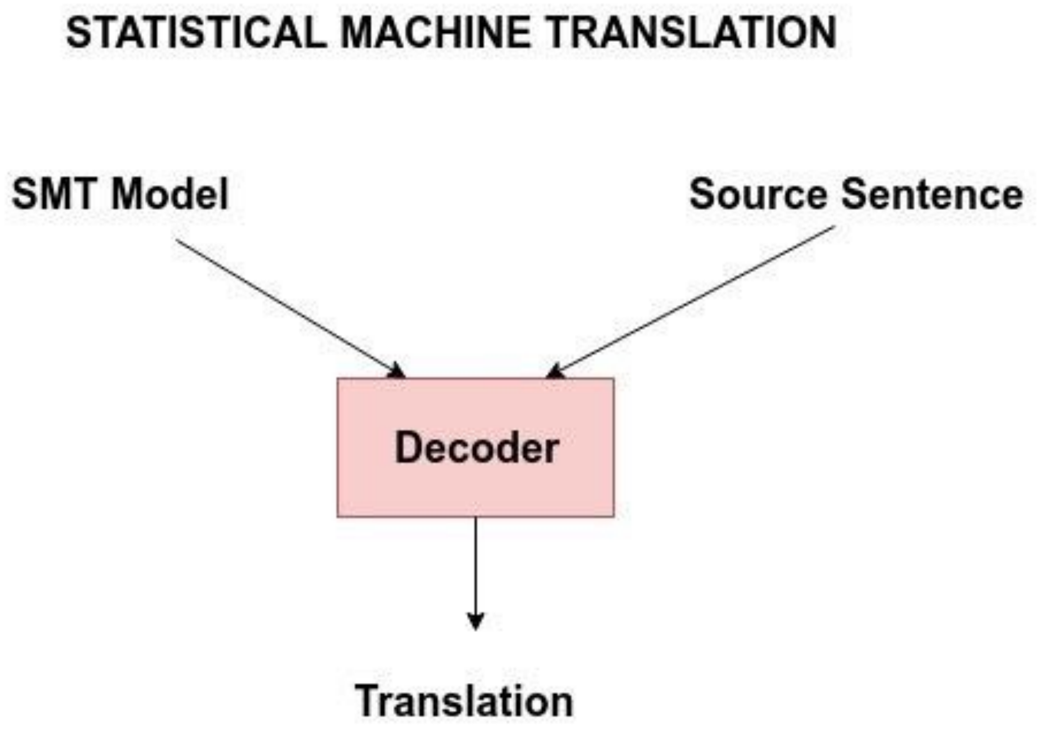
```
$ git clone https://github.com/moses-smt/mgiza.git
$ cd mgiza/mgizapp
$ cmake .
$ make
$ make install
$ manual-compile/compile.sh
```

3.2 METHODOLOGY

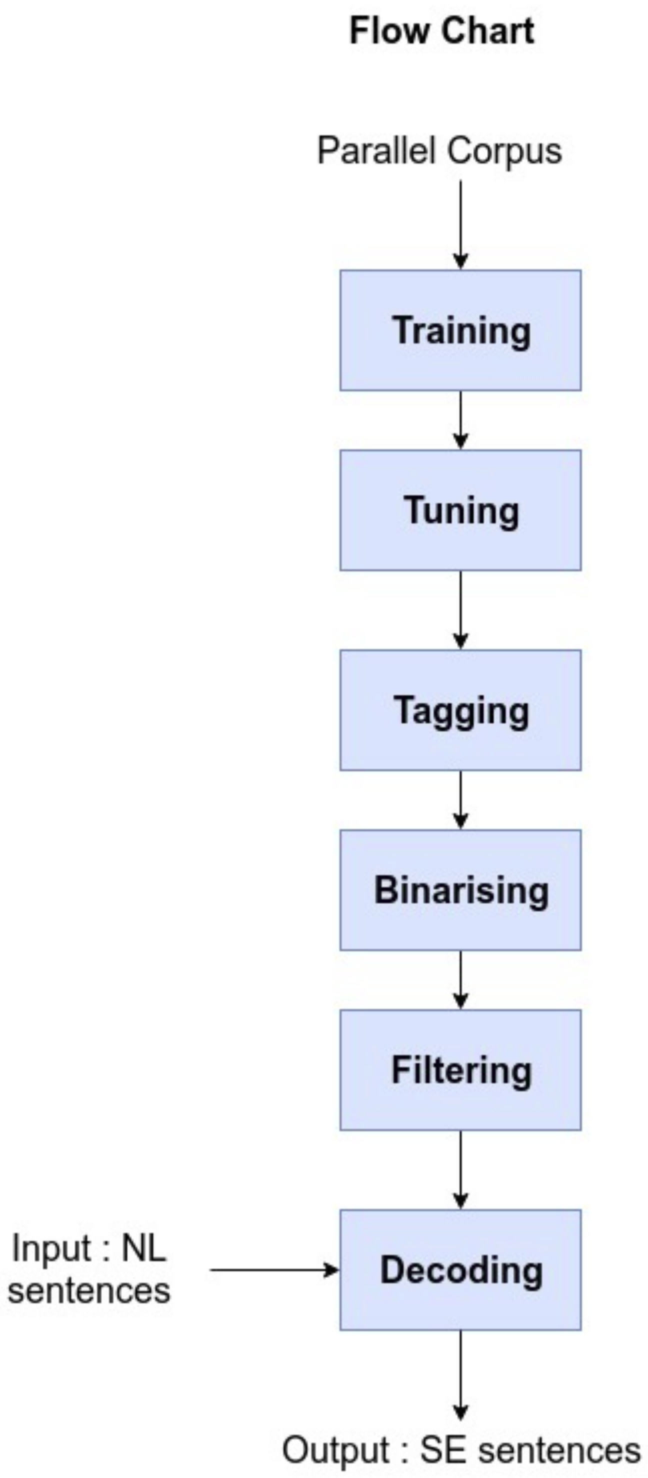
Phrase based Translation

In phrase based translation any linguistic information of input or output language is not needed. Most competitive statistical machine translation systems like CMU, IBM, ISI, and Google systems use phrase translation. In should be used when both input and output language have same structure. NL and SE are both english languages. Hence, it is used.

For translating any NL sentence to SE sentence , SMT Model and source NL sentence are needed. The decoder does the work of translating NL sentence to SE sentence and gives it as output (translation).



Generation of SMT Model :



For generation of SMT model or for Moses training, Parallel corpus of NL and SE sentences are needed. They go through following steps.

Parallel Corpus Preprocessing :

There are various parallel and monolingual corpora available in machine readable format. However, since SE has not be standardised yet, in this project, small NL-SE bilingual data is created.
Split paragraphs into sentences such that a nth sentence in the NL file corresponds to the nth sentence in the SE file.
Moses has inbuilt preprocessing methods.
The preprocessing steps involves checking whether the number of NL sentences is equal to the number of SE sentences. If they are not equal, then it will show failure at this step and will not proceed further.
Moses tokenises the text and converts the tokens into a standard case (lowercase).
Sentence pairs which are misaligned are removed.
Also, very long or very short sentences are removed.

Note :
FILENAME.en is NL file.
FILENAME.fr is SE file

To tokenise the data.

```
$ ~/mosesdecoder/scripts/tokenizer/tokenizer.perl -l en <
~/corpus/FILENAME.en > ~/corpus/FILENAME.tok.en
$ ~/mosesdecoder/scripts/tokenizer/tokenizer.perl -l fr <
~/corpus/FILENAME.fr > ~/corpus/FILENAME.tok.fr
```

To convert data to standard case (lowercase).
If this conversion is not done, Moses considers “Data” and “data” as two different words.

```
$ ~/mosesdecoder/scripts/recaser/train-truecaser.perl --
model ~/corpus/truecase-model.en --corpus ~/corpus/
FILENAME.tok.en
$ ~/mosesdecoder/scripts/recaser/train-truecaser.perl --
model ~/corpus/truecase-model.fr --corpus ~/corpus/
FILENAME.tok.fr
```

To delete sentences from parallel corpus which have size smaller than 1 or larger than 80

```
$ ~/mosesdecoder/scripts/training/clean-corpus-n.perl ~/
corpus/FILENAME.true fr en ~/corpus/FILENAME.clean 1 80
```

Training :

A phrase based translation table is created from the stored NL-SE phrase translation pairs.
Illustrated below are few entries of translations taken from phrase table:

```
becomes the rental car owner ||| must become the rental car
owner ||| 1 0.91608 0.333333 0.023181 ||| 0-1 1-2 2-3 3-4
4-5 ||| 1 3 1 ||| |||
is permitted that ||| may ||| 0.2 0.00575327 1 0.270707 |||
0-0 1-0 2-0 ||| 5 1 1 ||| |||
( the service mileage of the rental ||| ( the service
mileage of the rental ||| 1 0.938446 1 0.700471 ||| 0-0 1-1
2-2 3-3 4-4 5-5 6-6 ||| 1 1 1 ||| |||
is prohibited that a rental is open ||| No open ||| 0.5
1.11452e-06 0.166667 0.160494 ||| 1-0 6-1 ||| 2 6 1 ||| |||
```

In first entry “1 0.91608 0.333333 0.023181” , four scores are the inverse phrase translation probability, inverse lexical weighting, direct phrase translation probability and direct lexical weighting.

Tuning :

Tuning is a process of finding optimal weights for translation models which the decoder uses.

Algorithm of Tuning :

The tuning set consists of NL-SE parallel corpus. Tuning set should be different from training corpus. Otherwise, it will get over trained on that set and might not work well for unseen data. The tuning NL is decoded to get output SE, then the model weights are iteratively updated based on error calculated in the output SE. This iterative process continues until a predefined threshold is not crossed. The error metric is calculated using BLEU.

Algorithm of BLEU (bilingual evaluation understudy):

It is an algorithm for evaluating the quality of NL-SE translation. Quality depends on the difference between output SE and that of tuning set SE.

Example for 1 gram :

Reference data is tuning set SE.

For all words “w” in output SE,

Total score = summation of [(max number of “w” in reference SE) / (length of sentence in reference text)]

Same formula can be extended to n-grams.

Total score can be [0,1]. If total score for translated text is more than threshold then those weights corresponding to this translation are chosen.

Note:

FILENAME_TUNING.tok.en is tokenized NL file.

FILENAME_TUNING.tok.fr is tokenized SE file

```
$ ~/mosesdecoder/scripts/training/mert-moses.pl ~/corpus/
FILENAME_TUNING.tok.fr ~/corpus/FILENAME_TUNING.tok.en ~/
mosesdecoder/bin/moses ~/lm/train/model/moses.ini --mertdir
~/mosesdecoder/bin/ &> ~/working/mert.out
```

Tagging :

With tagging, a block of text in NL can be substituted by a block of text in SE with a probability value. For tagging, a list of tags are prepared and added in the input NL file. After tagging, the input NL file gets converted to the output NL file as mentioned below :

To enable tagging (It is a search and replace problem) :

```
sed -i "s,${DATA_TO_BE_REPLACED},${ITS_TAG},g" ~/working/
input_file_tag.txt
```

Input NL file :
each operating company *must have* at least one insurer.
a rental *may be* open only if an estimated rental price of the rental is provisionally charged to a credit card that is in the name of the renter who is responsible for the rental.

Output NL file with tags :
each operating company *<np translation="has" prob="1">must have</np>* at least one insurer.
a rental *<np translation="is" prob="1">may be</np>* open only if an estimated rental price of the rental is provisionally charged to a credit card that is in the name of the renter who is responsible for the rental.

Structure of Tag :

Tag : must not|<np translation="is not" prob="0.7" >must not</np>
Explanation : Above tag replaces “must not” with “is not” with probability 0.7.

Tag : must not|<np translation="_" prob="0.9" >text to be deleted</np>
Explanation : To replace a text in NL with null text, an underscore or similar character is used and the underscore is later removed from the SE.

Binarising :

If the models without binarising are used directly, then process of loading the phrase table is very slow. Therefore, it is converted to binary format which is easy to load.

Create a folder to store model in binarised form

```
$ mkdir ~/working/binarised-model

$ ~/mosesdecoder/bin/processPhraseTableMin -in ~/lm/train/
model/phrase-table.gz -nscores 4 -out ~/working/binarised-
model/phrase-table

$ ~/mosesdecoder/bin/processLexicalTableMin -in ~/lm/
train/model/reordering-table.wbe-msd-bidirectional-fe.gz -
out ~/working/binarised-model/reordering-table
```

Settings to run binarised model :

Copy moses.ini generated by tuning with proper weights into binarised-model folder

```
$ cp ~/working/mert-work/moses.ini ~/working/binarised-
model/
```

Change the phrase and reordering tables to point to the binarised versions in moses.ini in binarised-model folder, as follows:

1. Change PhraseDictionaryMemory to PhraseDictionaryCompact
2. Set the path of the PhraseDictionary feature to point to $HOME/working/binarised-model/phrase-table
3. Set the path of the LexicalReordering feature to point to $HOME/working/binarised-model/reordering-table

Filtering :

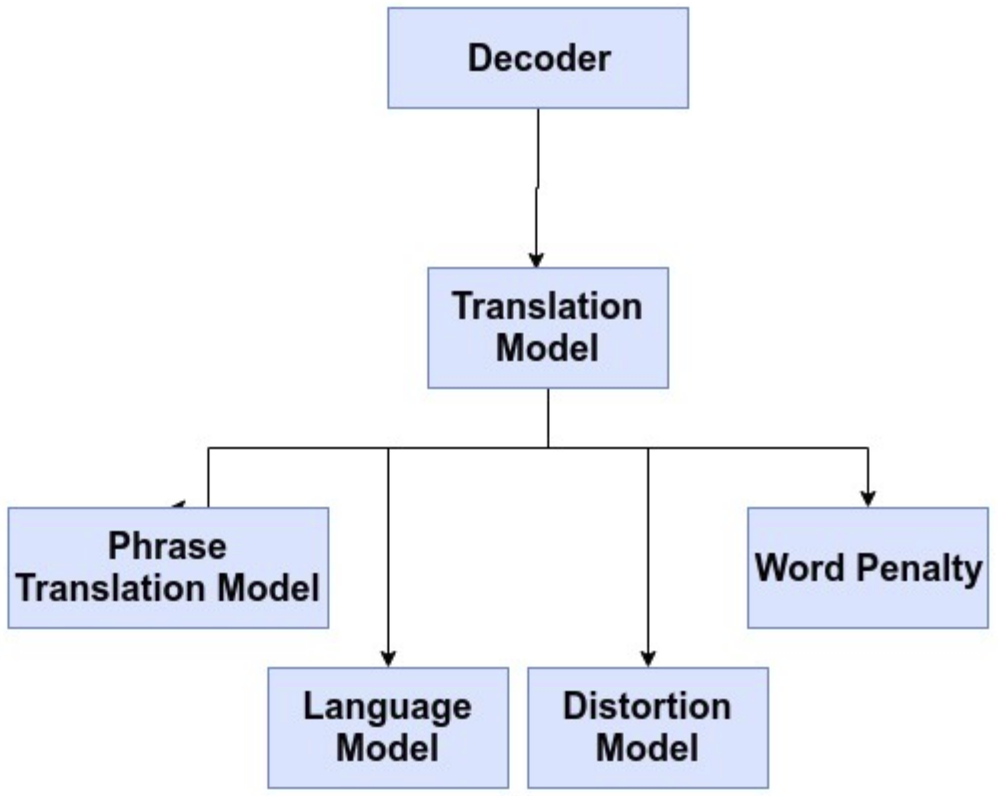
Phrase tables are very big, but for the translation of input SE only a fraction of the table is required. Hence, filtering is done to filter the translation table.

Note :
FILENAME.tok.en is input NL tokenised file.

To filter and binarise these tables, the script is :

```
$ ~/mosesdecoder/scripts/training/filter-model-given-
input.pl filtered mert-work/moses.ini ~/corpus/
FILENAME.true.en -Binarizer ~/mosesdecoder/bin/
processPhraseTableMin
```

Decoder :



Decoder finds highest scoring sentence in the SE language which corresponds to a given NL sentence. A beam search algorithm finds the highest probability translation among all the possible number of choices.

Beam Search Algorithm :

Beam search is based on heuristic, it only expands the most optimistic node. It reduces its memory space requirement by keeping only predetermined number of best partial solutions in the memory.

Future costs are computed for all contiguous spans over the sentence:

Example :

future cost from 0 to 0 is -9.875

future cost from 0 to 1 is -7.02

future cost from 0 to 2 is -2.72

future cost from 0 to 3 is -43.25

future cost from 1 to 1 is -2.78

future cost from 1 to 2 is -37.47

future cost from 1 to 3 is -82.128

future cost from 2 to 2 is -5.115

future cost from 2 to 3 is -51.26

future cost from 3 to 3 is -11.21

Total cost = Present cost + future cost.
It expands state with minimum total cost.

The decoder allows the user to customise the input structure. It can be a plain sentence or it can be annotated with xml-like elements or it can be a more complex structure like a lattice or a confusion network.

Translation model:

Translation models are trained from parallel corpora. It maps NL to SE in different ways.

Following are the four models used :

Phrase translation :

The translation systems cannot store all native strings of the input NL language and their translations in the output SE language. Hence, phrase translation table is formed during the training phase of the Moses flow chart. The phrase translation table ensures that the NL phrases and the SE phrases are good translations of NL-SE.

To speed up the decoder, search space can be limited by reducing the number of translation options available for each input NL phrase. The number of translation options possible depend on the number of entries in the phrase translation table. Therefore, fix the limit on translation options retrieved for each input phrase by using phrase translation probability threshold.

Language model :

Language models are typically approximated by *n*-gram models. It ensures that the output is fluent English.

N-Gram Probability Calculations:

$$\text{Bigram: } \Pr(a \mid b) = \frac{\Pr(ab)}{\Pr(a)}$$

$$\text{Trigram: } \Pr(ab \mid c) = \frac{\Pr(abc)}{\Pr(ab)}$$

$$\text{Quadgram: } \Pr(abc \mid d) = \frac{\Pr(abcd)}{\Pr(abc)}$$

Notation :

Pr(X) is probability of getting word “X” in text.

Pr(a|b) is probability of getting word “b” after word “a” if word “a” is present in text.

Pr(ab|c) is probability of getting word “c” after word “ab” if word “ab” is present in text.
Pr(abc|d) is probability of getting word “d” after word “abc” if word “abc” is present in text.
Similarly, for N gram probability.

Different Language Models like SRILM, IRSTLM, KenLM and RandLM are available.

SRILM has been chosen because it is recommended to use SRILM as it is also used in Ubuntu NLP Repository. It is written in C++. It produces n-gram word counts in ARPA format which is standard in NLP communities. It provides compact representation that enables efficient utilization of the language model.

Distortion model D(e,f) (weight-d)
Distortion model decides the amount reordering of words allowed. Reordering can be limited to a maximum number of words skipped with the switch -distortion-limit, setting this parameter to 0 means no reordering and -1 means unlimited reordering. This value is changed and results are obtained accordingly.

Word penalty W(e) (weight-w)
The word penalty ensures that the translations are not too long or short. Negative value of W(e) gives longer output. This value is changed and results are obtained accordingly.

Working of decoder:

The total cost of translation is:

$$p(e|f) = phi(f|e)^{weight_{phi}} * LM(e)^{weight_{lm}} * D(e,f)^{weight_{d}} * W(e)^{weight_{w}}$$

where,

e : NL

f : SE

p(e|f) : total cost of translating e to f.

weight-phi (Phrase translation model), weight-l (Language model), weight-d (Distortion model), weight-w (Word penalty) are model weights obtained in tuning.

phi(f|e) : Phrase translation model score

LM(e) : Language model score

D(e,f) : Distortion model score

W(e) : Word penalty score

The decoder finds the highest scoring sentence in the SE language which corresponds to a given NL sentence using above cost function.
These weights are present in the configuration file moses.ini.

To run the decoder:
INPUT_FILENAME is NL file to be translated to SE.
OUTPUT_FILENAME is SE file generated.

```
$ ~/mosesdecoder/bin/moses -xml-input exclusive -f ~/
working/binarised-model/moses.ini < INPUT_FILENAME.txt >
OUTPUT_FILENAME.txt
```

“-xml-input exclusive” flag is added because tags are used.
Moses.ini has all the configurations and weights needed for translation of text from NL to SE.
The decoder keeps a stack of the n best translations, to increase speed limit value n=100 is used.

3.3 DRAWBACKS

Phrase-based models :
Phrase based machine translation can capture translation information that can fit in n-gram model (less than n words phrases). If n is more than 5, it will have high computation time. As an example, consider NL to SE translation, phrase-based translation cannot translate longer phrases. Hence, special structures are needed. Phrase Table rules formed are redundant.

3.4 DEBUGGING

1. While installing moses on Ubuntu 16.04.
Error: libtool is undefined
    http://stackoverflow.com/questions/18978252/error-libtool-library-used-but-libtool-is-undefined
Solution: Switch to ubuntu 14.04 lts.

2. While running following command to install boost:

```
./b2 -j4 --prefix=$PWD --libdir=$PWD/lib64 --layout=system link=static install || echo FAILURE
```

Error: dependencies errors

Solution: Compatibility issues with ubuntu 16.04. Switch to Ubuntu 14.04 lts

3. While executing a sh script

Error: Syntax error: bad fd number

Solution: http://stackoverflow.com/questions/15809060/sh-syntax-error-bad-fd-number

3.5 RESULTS

Input Sentences :

1. The fuel level of the rented car of a rental must be full at the actual pick-up date-time of the rental.
2. A rental may be open only if an estimated rental price of the rental is provisionally charged to a credit card that is in the name of the renter who is responsible for the rental.
3. A rental must incur a location penalty charge if the drop-off branch of the rental is not the return branch of the rental.

Case - 1

Model Weights used :

Distortion0= 0.0160308

LM0= 0.0295526

WordPenalty0= 0.0133585

PhrasePenalty0= 0.032576

TranslationModel0= 0.0109651 0.00166967 0.0147672 0.00524506

UnknownWordPenalty0= 1

Output :

1. the rented fuel level that the rented rented car that a rental is full at the rented actual pick-up date-time that the rented rental.
2. a rental is open only if only a an that the rented rental estimated rental price of the rental is provisionally charged to a credit card that is included in the rented name that the rented renter who that is responsible for the rented rental.
3. a rental incurs a location to penalty charge a the rented drop-off branch that the rented car of the rental rental is not the rented return branch that the rented rental.

Explanation :

Language model weight is 0.0295526. Therefore, fluency is less.

Case - 2

Model Weights used :

Distortion0= 0.3

LM0= 0.5

WordPenalty0= -1

PhrasePenalty0= 0.2

TranslationModel0= 0.2 0.2 0.2 0.2

UnknownWordPenalty0= 1

Output :

1. the fuel level of the rented car of a rental is full at the actual pick-up date-time of the rental.
2. a rental is open only if an estimated rental price of the rental is provisionally charged to a credit card that is in the name of the renter who is responsible for the rental.
3. a rental incurs a location penalty charge if the drop-off branch of the rental is not the return branch of the rental.

Explanation :

Language model weight is 0.5. Therefore, fluency is more.

Case - 3

Model Weights used :

Distortion0= 0.0160308

LM0= 0.0295526

WordPenalty0= -1

PhrasePenalty0= 0.032576

TranslationModel0= 0.0109651 0.00166967 0.0147672 0.00524506

UnknownWordPenalty0= 1

Output :

1. the rented car of the rental fuel level of the the rented car of the rental rented car of the No the rental is full at the rented car of the rental actual pick-up date-time of the the rented car of the rental rental.
2. No the rental rented car of the rental is open only if only a an estimated the rental price of the the rented car of the rental the rental rented car of the rental is provisionally charged to a credit card that rented car of the rental is in the rented car of the rental name of the the rented car of the rental renter who rented car of the rental is responsible for the rented car of the rental rental.
3. No the rental incurs a location to a branch penalty charge a the rented car of the rental drop-off branch of the the rented car of the rental the rental rented car of the rental is not the rented car of the rental return branch of the the rented car of the rental rental.

Explanation :

Negative value of Word Penalty gives longer sentences.

NEURAL MACHINE TRANSLATION

4 . NEURAL MACHINE TRANSLATION

4.1 SELECTION AND INSTALLATION

Selection of Deep Learning for MT :

Machine Learning is extremely useful for tasks like regression and classification when good features of input data are available. These features are generally hand-crafted to suit the input. However, natural language is extremely complex and it is almost impossible to enumerate all the features. Thus, the main aim of machine learning is optimising weights on given features.

Deep Learning is a subfield of machine learning that enables the system to learn the features, given the input data. The features extracted by deep learning models may not be easily understandable.

The tool used to create and train deep learning models is “ **Tensorflow** “ .

Installation :

1. Installation of anaconda version 4.3.7
2. Creation of virtualenv with anaconda & python=3.5.1

```
$conda create -n FYPvenv python=3.5.1 anaconda
```

3. Installation of tensorflow into the virtualenv

```
$conda install -c conda-forge tensorflow
```

4. Installation of nltk

```
$sudo pip install -U nltk
```

4.2 SELECTION OF THE MT MODEL

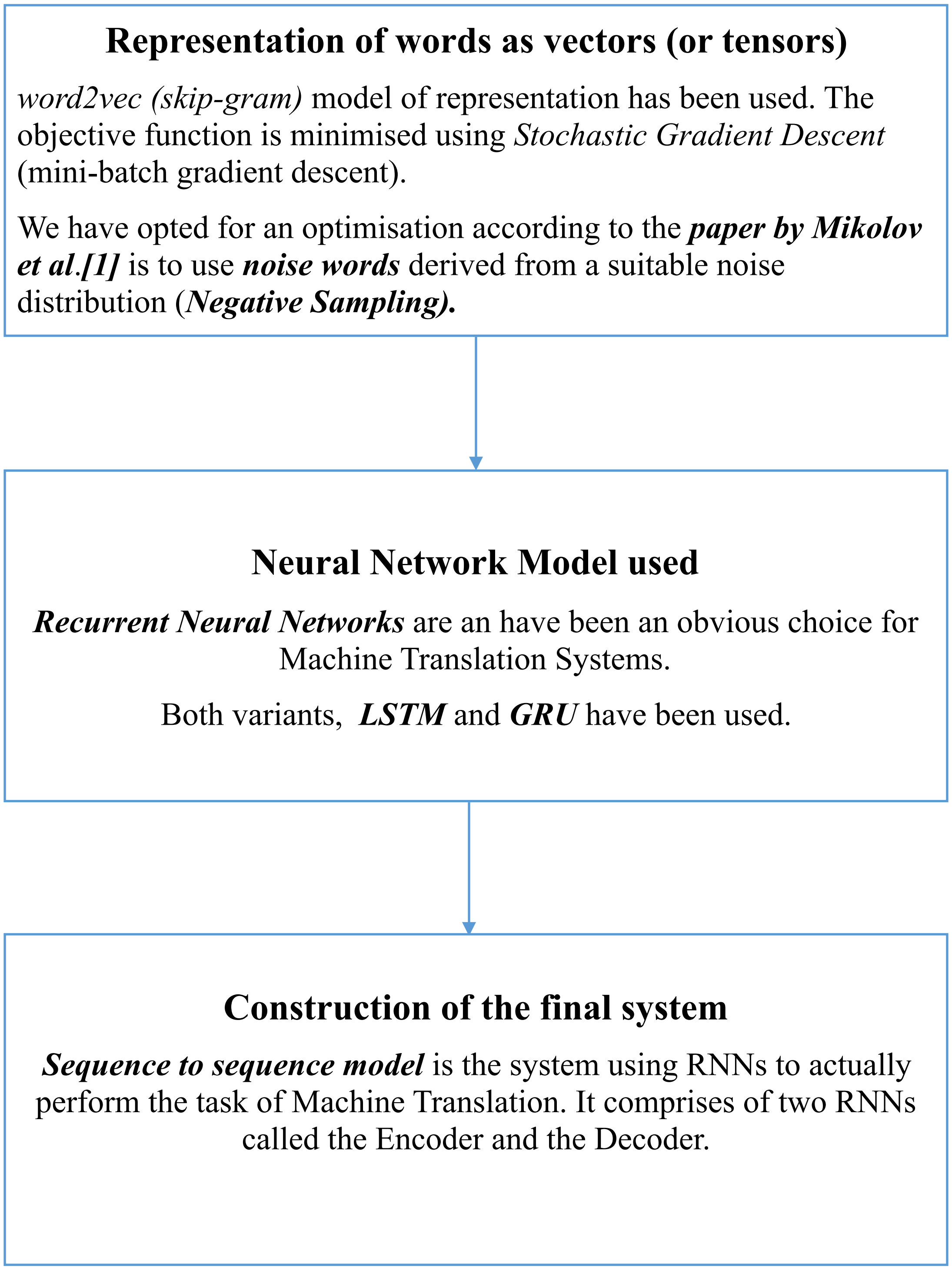
Selection of the NMT Model :

There are three main choices made while designing a NMT system :

1. Representing words as vectors (or tensors).
2. Deciding which Neural Network is to be used for language modelling.
3. Constructing a encoder-decoder system to derive relation between corresponding source and target vectors.

Given below, is an illustration of the selection of the above units, describing the pros and cons for each.

The appropriate justification is given in the section after.



Representation of words as vectors (or tensors)

word2vec (skip-gram) model of representation has been used. The objective function is minimised using *Stochastic Gradient Descent* (mini-batch gradient descent).

We have opted for an optimisation according to the ***paper by Mikolov et al.[1]*** is to use ***noise words*** derived from a suitable noise distribution (***Negative Sampling).***

Neural Network Model used

Recurrent Neural Networks are an have been an obvious choice for Machine Translation Systems.

Both variants, ***LSTM*** and ***GRU*** have been used.

Construction of the final system

Sequence to sequence model is the system using RNNs to actually perform the task of Machine Translation. It comprises of two RNNs called the Encoder and the Decoder.

Justification of the above choices :

<u>CHOICE OF REPRESENTATION OF WORDS (WORD2VEC):</u>

There are many approaches for representing words.
There are many problems with discrete representation of words (for eg: synonyms sets or as Directed Acyclic Graphs or assigning word Ids). Few are enlisted :

1. Discrete representations fail to capture nuances.
2. It is difficult for discrete representations to adapt to new words.
3. A lot of human labour is required to create and maintain such representations.
4. These representations are subjective.

Vector representations are extremely scalable and with proper representations, it is possible to capture the relationship between words.

There are various choices for vector representations.

1. **One-Hot word representation :** Represent each word as a one-hot vector. The number of dimensions of the vector is equal to the number of words in the vocabulary. The vector is constructed as a column vector of 0 with 1 in the index where it occurs in the vocabulary list.
   For example, in a vocabulary of 10 words,
   Cat = [1 0 0 0 0 0 0 0 0 0]
   Dog = [0 0 1 0 0 0 0 0 0 0]
   This representation has various shortcomings :
   i) It is not scalable. Adding words is difficult.

ii) It doesn't capture any kind of relation between words.

2. **Co-occurrence matrix :** The idea is to represent words using their neighbours instead of its own index. If the context is the entire document, it gives general topics and is useful for capturing syntax. A general method is to use a window based co-occurrence matrix (symmetric window/ window to left or right). It helps in capturing similarities. For example, input text : " I like light music " ; " I like dance " ; " I enjoy playing " , using window size = 1.

counts	I	LIKE	ENJOY	LIGHT	MUSIC	DANCE	PLAYING
I	0	2	1	0	0	0	0
LIKE	2	0	0	1	0	1	0
ENJOY	1	0	0	0	0	0	1
LIGHT	0	0	0	0	1	0	0
MUSIC	0	0	0	1	0	0	0
DANCE	0	1	0	0	0	0	0
PLAYING	0	0	1	0	0	0	0

This model does capture similarity. "like" and "enjoy" are similar. However, vocabularies are generally large and the matrix is very high dimensional. It is also not a scalable scheme**.**

A solution for the large dimensionality is to use Principal Component Analysis to reduce the size of the matrix.

3. **Word2vec :** Instead of capturing word counts, the aim is to predict the word and the surrounding words. This model was introduced by *Mikolov et al[1].* There are basically two approaches that can be used :

a. Skip gram model :

<u>*Basic Idea :*</u>

- This model predicts the surrounding words given the centre word. For example, consider the text

  w1 w2 <u>w3</u> w4 w5 w6

- Considering window size = 2, ***w3*** is the centre word and ***w1,w2*** and ***w4,w5*** are to be predicted. Once, this is done, move on to the next window as shown

  w1 ***w2 w3 <u>w4</u> w5 w6***

<u>*Algorithm*</u>

- Notation
  - m : window size
  - T : entire training set
  - w_i : word vector corresponding to i
  - u_i : word vector when i is the outside word
  - v_i : word vector when i is the centre word
- Note that each word has two vectors associated with it, one when it appears as the centre word (denoted by v) and one when it appears as a context/ surrounding word (denoted by u).
- Both vectors corresponding to all the words are initialised with random values taken from an uniform distribution between *[-1.0 to 1.0]*
- The Objective function is optimised to maximise the log probability of the word vector of the surrounding word, given the centre word as shown

$$J(\theta) = (1/T)\sum_{i=1}^{T} \sum_{-m<=j<=m; j\neq 0} (logP(w_{t+j}|w_j))$$

In the above equation, θ represents all the word vectors which are jointly optimised to maximise J.

- The probability can be expressed in terms of the softmax function as follows

$$p(o|c) = \frac{\exp\left(u_o^T v_c\right)}{\sum_{w=1}^{W} \exp\left(u_w^T v_c\right)}$$

In the above equation, *o* represents the outside/surrounding word and *c* represents the centre word. Thus, u_o represents the word vector corresponding to the surrounding word, *o,* when it appears on the outside and v_c represents the word vector corresponding to the centre word, c, when it appears as the centre word.

- The optimisation of the objective function, J is done using a variant gradient descent, called Stochastic gradient descent. The idea is shown below :

```
repeat until convergence for θ
    $\theta_{new} = \theta_{old} - \alpha.\partial J/\partial\theta_{old}$
```

- The final word vector w_i for word i is given as, w_i = *average(*u_i*,* v_i*)*

b. Continuous Bag of Words model : A variation of the above, this model predicts the centre word, given the surrounding words (in a window).

- In this project, ***word2vec (skip-gram model) representation has been used***. The objective function is minimised using Stochastic Gradient Descent (mini-batch gradient descent).

- Another optimisation done is according to the ***paper by Mikolov et al***. is to use ***noise words*** derived from a suitable noise distribution. The objective function is suitably modifies to increase probability when the correct words are used and reduces the probability when noise words are the centre words.

```
I had a sweet apple.
```

- With window size = 1, the (context, target) pairs are :

```
( [I , a] , had) , ( [had , sweet] , a ) , ( [a , apple] , sweet)

  (had , I ) , ( had , a ), ( a , had ) , ( a , sweet ) ...
```

Consider a noise word, say ‘*zebra*’.

Now the objective function can be modified as the following :

$$J(\theta) = log(P(D = 1|sweet, apple)) + log(P(D = 0|zebra, apple))$$

CHOICE OF NEURAL NETWORK

The first and most important task in any machine translation system is language modelling.

A language model can be thought of as the probability of a sequence of words occurring together. Valid sentences have a higher probability as compared to random sentences. Mathematically it is :

$$P(w_1, w_2, ..., w_i, ..., w_n)$$

It is intuitive that, P (I had a sweet apple) > P (had sweet I a apple)

Traditional MT systems work on Markov assumption and consider only a window on n words.

Probabilities of all unigrams, bigrams and n-grams are to be stored for good results as shown.

$$P(w_1, w_2, ..., w_n) = \prod_{i=1}^{n} P(w_i|w_{(i-n+1)}....w_{(i-1)})$$

Each n-gram is calculated in terms of their counts and stored that is,

$$p(w_2|w_1) = \frac{\text{count}(w_1, w_2)}{\text{count}(w_1)} \qquad p(w_3|w_1, w_2) = \frac{\text{count}(w_1, w_2, w_3)}{\text{count}(w_1, w_2)}$$

However, this demands a very huge RAM requirement and storage capacity.

- A solution to this is ***using Recurrent Neural Networks*** where the memory requirement scales with the number of words.

Basic Idea :

A recurrent neural network has the ability to remember all the previous information fed. In other words, recurrent neural networks are persistent. This is because they have loops in them.

In general, a RNN is conditioned on all the previous words hence the incorrect yet essential Markov assumption in Traditional MT systems is not required anymore.

Working :

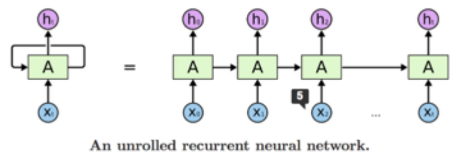


Figure 1 : Network Model

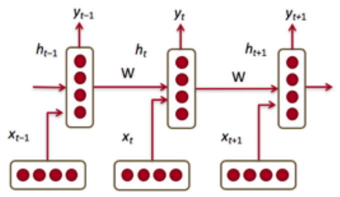


Figure 2 : Detailed view

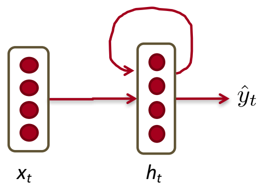


Figure 3 : A single step

IMAGE TAKEN FROM HTTP://COLAH.GITHUB.IO/POSTS/2015-08-UNDERSTANDING-LSTMS/

- Figure 1 shows the network model.
- Figure 2 shows the detailed view of the model.
- Figure 3 shows a single step of the process.

- Notation : $(x_1, x_2, \ldots, x_{t-1}, x_t, x_{t+1}, \ldots, x_T)$: the list of word vectors

  x_t : word vector fed to the network at time step t

  h_t : the hidden layer at time step t

  W_{hx} : Weights between the hidden layer and the input layer

  W_{hh} : Weights between hidden layer h_{t-1} and h_t

  $\hat{y}_t$: Output class (i.e. the next word)

  σ : sigmoid function

  W : weights for the softmax classifier

- *At each step,*

  $$h_t = \sigma (W_{hx} x_t + W_{hh} h_{t-1})$$

  $$\hat{y}_t = softmax(W. h_t)$$

- The errors are then back propagated through the network for computation of new weights.
- $\hat{y} \in \mathbb{R}^{|V|}$ is thus, the probability distribution over the vocabulary.

- Note that since same weights are used for computation at each time-step, this model is scalable.

Though having the same weights at each stage makes RNNs scalable, it brings with it a problem called “Vanishing or Exploding gradient problem”.

RNNs are able to “remember” all the previous information stored because of back-propagation of error from an output unit to all the previous hidden units. However, it is empirically observed that this does not happen after about seven steps.

This can be reasoned in the following manner.

- In a gradient based training model, if the gradient value is large, the training is faster and vice-versa.
- The gradient of the first few layers in a deep neural network are small. This is because, the first few layers are designed to understand simple patterns.
- This results in a low gradient value. When error back propagates through these layers, the gradient value becomes so small that there is no effective training.

Hence, when the gap upto which information is to be remembered is large, RNNs fail.

The solution to this problem comes in the form of modified RNNs - LSTMs and GRUs. Both these techniques have been used for comparative analysis in this project.

Gated Recurrent Units (GRU) :

Basic Idea :

Gated Recurrent units allow memories to capture long distance relationships between words.

Example for illustration:

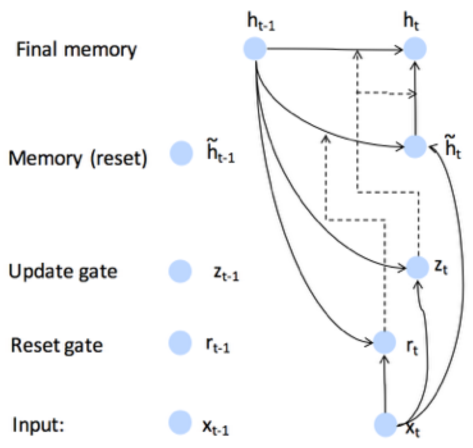
- Consider the example of a paragraph about the war between India and Pakistan. Suppose that there is a mention of the countries at the start of the paragraph, followed by a long description of the techniques used in the war. Finally, there is a concluding statement that says, the war took place at ___ . Here it is obvious that the blank is filled with a place from India / Pakistan. If this to be captured by the computer, it has to deduce long term dependencies and relationships.

The main modification in GRUs is that it allows errors to flow at different rates to different units. This implies that errors are propagated with different strengths to different units depending on the input. GRUs use gates for this purpose. Gates are either open(on)/closed(off). The gates corresponding to units which have short distance dependencies are mostly ON. Gates are implemented as a hidden layer of sigmoid units.

Working :

Each of the components is explained below :

- Reset gate(r_t) and Update gate(z_t) are computed as sigmoid of simple linear transformations of the input and the hidden layer.
- The new memory, $\tilde{h}_t$ has an element-wise product of reset gate and hidden layer in the previous time-step. When r=0, only the current time-step is considered and the previous time step is ignored.
- The final updation uses the value of the update gate as well as the new memory. If the update gate (z) = 1, then the input at this time step is completely ignored and the previous step is copied over. This direct copying over allows the error to propagate without "vanishing".



$$z_t = \sigma\left(W^{(z)}x_t + U^{(z)}h_{t-1}\right)$$
$$r_t = \sigma\left(W^{(r)}x_t + U^{(r)}h_{t-1}\right)$$
$$\tilde{h}_t = \tanh\left(Wx_t + r_t \circ Uh_{t-1}\right)$$
$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t$$

CREDITS ; : RICHARD SOCHER

Long Short Term Memories (LSTM) :

Basic Idea :

LSTMs are similar to GRUs but have more complex units than GRUs.

LSTM allows the unit to retain information (in its memory cell) without exposing it to the entire network.

LSTMs are stacked for better results.

Working :

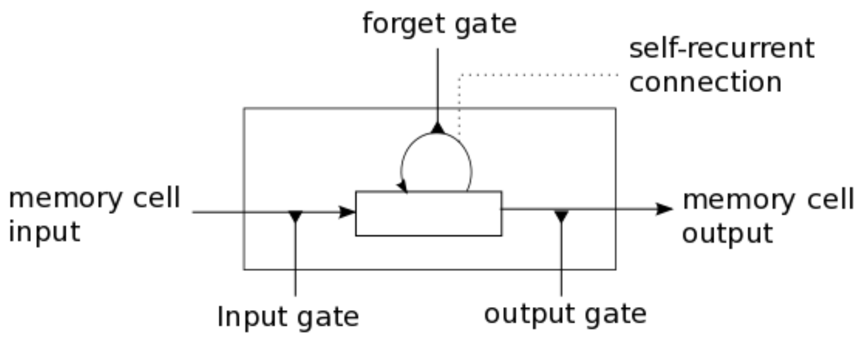


IMAGE CREDITS : DEEPLEARNING.NET

- There are THREE gates as shown :
  - Input gate : $i_t = \sigma(W^{(i)}x_t + U^{(i)}h_{(t-1)})$, this shows how much the current input matters.
  - Forget Gate : $f_t = \sigma(W^{(f)}x_t + U^{(f)}h_{(t-1)})$, which is used to forget the past (if gate=0).
  - Output Gate : $o_t = \sigma(W^{(o)}x_t + U^{(o)}h_{(t-1)})$, which determines how much of the memory cell is exposed to the network.

$\tilde{c}_t$

- New Memory cell , $= tanh((W^{(c)}x_t + U^{(c)}h_{(t-1)})$
- Final Memory cell , $c_t = f_t \circ c_{t-1} + i_t \circ c_t$
- Final hidden state , $h_t = o_t \circ tanh(c_t)$

CHOICE OF FINAL MT SYSTEM

Sequence to sequence model is the system using RNNs to actually perform the task of Machine Translation.

It comprises of two RNNs called the Encoder and the Decoder. The encoder and decoders are different RNNs and have different weights, though it is possible for them to share weights.

A sampled softmax function aids the process. For a large output library it helps in keeping track of the output projection. The larger the vocabulary variations of the standard softmax functions minimise the required effort.

The model makes use of bucketing to handle the sentences which vary in length. The input in NL can have different length sentences on the output generated in SE. There are two alternatives either by generating a *seq2seq* model for each pair in NL and SE, which would result in a large graph consisting of smaller subgraphs. The other way is to pad a sentence with special symbols. This toughens life when it comes to sentences of small length. The tool instead provides for different buckets in which the sentences can be classified into and required minimum possible padding is done. The pairs of buckets depend on the general analysis of the translation pairs in NL to SE. The parameterised bucketing available makes it more functional in this real life application of varied length sentence translations. The default bucket pairs are modified according to the need.

ERROR MEASURE :

The error measure used here is the perplexity. Perplexity is the measure of how "confused" the model in prediction of words.

Mathematically, Perplexity, where H is the cross-entropy error is defined as

$$P = 2^H$$

4.3 EXPERIMENTS

CREATION OF DATASET

A parallel corpus for NL-SE containing 40 sentences was created manually.

The SE sentences corresponding to EU-Rent were inspired by *Annex-G* of the SBVR Documentation.

CHOICE OF HYPER-PARAMETERS

The following hyper-parameters were considered for the experiments

1. Type of cell : Either ***GRU*** or ***LSTM*** cells.
2. Power of the model : Structure of the neural network including
    1. Number of layers
    2. Number of units per layer
3. Training time : Measured by the number of iterations

#	Neuron units	seq2seq	Input	Output	Perplexity	Remarks
1a	GRU	100 units per layer 3 layers 100 iterations	1. A renter may be responsible for more than one advance rental. 2. No open rental may authorize a driver who is barred. 3.Each operating company must have at least one insurer. 4. A rental may be open only if an estimated rental price of the rental is provisionally charged to a credit card that is in the name of the renter who is responsible for the rental. 5.If an estimated rental price of a rental is provisionally charged to a credit card that is in the name of the renter who is responsible for the rental then it is permitted that the rental is open.	1. It is _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK 2. It _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK 3. is _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK 4. It is _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK 5. It is _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK	22.90	Most basic model used for testing. Since, the number of units are less in each layer, perplexity is low even with low number of iterations. However, the model does not understand much. From the entire training set, the only information gathered is that a sentence generally starts with It is in SE.

#	Neuron units	seq2seq	Input	Output	Perplexity	Remarks
1b	LSTM	100 units per layer 3 layers 100 iterations	1. A renter may be responsible for more than one advance rental. 2. No open rental may authorize a driver who is barred. 3.Each operating company must have at least one insurer. 4. A rental may be open only if an estimated rental price of the rental is provisionally charged to a credit card that is in the name of the renter who is responsible for the rental. 5.If an estimated rental price of a rental is provisionally charged to a credit card that is in the name of the renter who is responsible for the rental then it is permitted that the rental is open.	1. It is is is is is is is is is is is is is is 2. It is is is is is is is is is is is is is is 3. It is is is is is is is is is is rental rental rental 4. It is is is is is is is is is is is is is is 5. It is is is is is is is is is is is is is is	21.17	Still very basic. Since an LSTM cell is by itself more complicated than a GRU cell, few other words, except it and is has been introduced and UNK tokens have been removed. The perplexities are very similar.

#	Neuron units	seq2seq	Input	Output	Perplexity	Remarks
2a	GRU	100 units 3 layers 500 iterations	1. A renter may be responsible for more than one advance rental. 2. No open rental may authorize a driver who is barred. 3.Each operating company must have at least one insurer. 4. A rental may be open only if an estimated rental price of the rental is provisionally charged to a credit card that is in the name of the renter who is responsible for the rental. 5.If an estimated rental price of a rental is provisionally charged to a credit card that is in the name of the renter who is responsible for the rental then it is permitted that the rental is open.	1.It is prohibited that a estimated is is open open is is is charged charged credit credit is is is 2.It is prohibited that a rental is open open a driver is is for for rental is is 3. is is that the _UNK _UNK _UNK _UNK _UNK _UNK _UNK is is is _UNK 4.It is obligatory that an estimated price price price open open is is provisionally charged to credit credit card is is in the name name renter renter who is responsible for the rental . 5. It is obligatory that an estimated price price of an open open is is charged to credit credit card is is in the name name renter renter renter who responsible responsible for the .	3.51	

#	Neuron units	seq2seq	Input	Output	Perplexity	Remarks
2b	LSTM	100 units 3 layers 500 iterations	1. A renter may be responsible for more than one advance rental. 2. No open rental may authorize a driver who is barred. 3.Each operating company must have at least one insurer. 4. A rental may be open only if an estimated rental price of the rental is provisionally charged to a credit card that is in the name of the renter who is responsible for the rental. 5.If an estimated rental price of a rental is provisionally charged to a credit card that is in the name of the renter who is responsible for the rental then it is permitted that the rental is open.	1.It is is is is is is is rental0 rental0 _UNK _UNK _UNK . 2.It is prohibited that _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK . . 3.It is permitted the the _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK 4.It is obligatory that an estimated rental price an an open is is provisionally charged to credit credit card card in in name name name renter renter renter who responsible responsible responsible 5. It is obligatory that an estimated estimated price an an open is is provisionally charged to credit credit card card in in name name name renter renter renter who responsible responsible for	4.79	A lot of iterations on a less powerful model does not help greatly.

#	Neuron units	seq2seq	Input	Output	Perplexity	Remarks
3a	GRU	256 units 3 layers 100 iterations	1. A renter may be responsible for more than one advance rental. 2. No open rental may authorize a driver who is barred. 3.Each operating company must have at least one insurer. 4. A rental may be open only if an estimated rental price of the rental is provisionally charged to a credit card that is in the name of the renter who is responsible for the rental. 5.If an estimated rental price of a rental is provisionally charged to a credit card that is in the name of the renter who is responsible for the rental then it is permitted that the rental is open.	1. It obligatory obligatory that rental 2. obligatory that that that rental rental rental rental rental rental rental rental rental rental rental 3. the the that that rental rental rental rental rental rental rental the the the the 4. It obligatory that that that rental 5. It obligatory that that that rental	717.80	Since the model is complex, it is not able to learn much when number of iterations are small. Has learnt that obligatory comes in most sentences. Perplexity is quite high since number of iterations is low.

#	Neuron units	seq2seq	Input	Output	Perplexit y	Remarks
3b	LSTM	256 units 3 layers 100 iterations	1. A renter may be responsible for more than one advance rental. 2. No open rental may authorize a driver who is barred. 3.Each operating company must have at least one insurer. 4. A rental may be open only if an estimated rental price of the rental is provisionally charged to a credit card that is in the name of the renter who is responsible for the rental. 5.If an estimated rental price of a rental is provisionally charged to a credit card that is in the name of the renter who is responsible for the rental then it is permitted that the rental is open.	1.It is is is is is rental rental rental rental is is is is is is is is is 2.It It is that that that rental rental rental rental rental is is is is is is 3. It is is that that rental rental rental rental rental rental rental rental rental rental 4. It is is is is is is is is is is is is is is is is is is is is is is is is 5.It is is is is is is is is is is is is is is is is is is is is is is is is	18.28	Model still needs time to understand.

#	Neuron units	seq2seq	Input	Output	Perplexity	Remarks
4a	GRU	512 units per layer 3 layers 100 iterations	1. A renter may be responsible for more than one advance rental. 2. No open rental may authorize a driver who is barred. 3.Each operating company must have at least one insurer. 4. A rental may be open only if an estimated rental price of the rental is provisionally charged to a credit card that is in the name of the renter who is responsible for the rental. 5.If an estimated rental price of a rental is provisionally charged to a credit card that is in the name of the renter who is responsible for the rental then it is permitted that the rental is open.	1.It It obligatory obligatory a a _UNK for of of of _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK of of of _UNK _UNK 2.It the the the the _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK the the the the _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK 3. It the the the the the _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK the 4. It It obligatory a a a a an an of the the the _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK of of of of _UNK a a a obligatory _UNK _UNK _UNK _UNK of of of of _UNK a a a obligatory _UNK for _UNK _UNK _UNK _UNK _UNK 5. It It an an an an an an open of of rental rental a a _UNK _UNK _UNK _UNK _UNK _UNK of of _UNK _UNK _UNK a a obligatory the _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK	416.53	Now the neural network is even more complex. Note that even though the perplexity is lower than when 256 units were used, UNK tokens are introduced. Few of the starting words of the sentences has been predicted correctly. This calls for training the model on higher number of iterations.

#	Neuron units	seq2seq	Input	Output	Perplexity	Remarks
4b	LSTM	512 units per layer 3 layers 100 iterations	1. A renter may be responsible for more than one advance rental. 2. No open rental may authorize a driver who is barred. 3.Each operating company must have at least one insurer. 4. A rental may be open only if an estimated rental price of the rental is provisionally charged to a credit card that is in the name of the renter who is responsible for the rental. 5.If an estimated rental price of a rental is provisionally charged to a credit card that is in the name of the renter who is responsible for the rental then it is permitted that the rental is open.	1.is the the the the rental rental rental rental is is is the the the the rental rental rental rental the the the the the 2. is is the the the rental rental rental rental rental rental rental rental the the the the the the the the the the the the 3. It is is that that that rental rental a a a a a a is is is is is is is is the the the the the the the is is the the the the the the the the the the the the the the the the the the the the 4. It is is that that that rental rental a a a a a a is is is is is is is is the the the the the the the is is the the the the the the the the the the the the the the the the the the the the 5. It It prohibited that that that an an a a a a a is is is is is is is is is is the is is is is is is the the the the the the the the the the the the the the the the the the the the	29.43	This is the result of training a highly complex model for very few iterations. Though perplexity is less, results are not that great.

#	Neuron units	seq2seq	Input	Output	Perplexity	Remarks
5a	GRU	512 units per layer 3 layers 500 iterations	1. A renter may be responsible for more than one advance rental. 2. No open rental may authorize a driver who is barred. 3.Each operating company must have at least one insurer. 4. A rental may be open only if an estimated rental price of the rental is provisionally charged to a credit card that is in the name of the renter who is responsible for the rental. 5.If an estimated rental price of a rental is provisionally charged to a credit card that is in the name of the renter who is responsible for the rental then it is permitted that the rental is open.	1. It is prohibited that the the is is rental0 a a is who additional additional who is is authorized another rental 2. It is prohibited that the _UNK of of rental rental is is is is for rental rental rental . 3. It is that the _UNK _UNK _UNK that is is in a _UNK _UNK is 4. It is obligatory that an estimated rental price of an open open is is provisionally to to a credit card is is is the name the the renter who is responsible responsible the rental rental . . 5. It is obligatory that an estimated rental price of an open open is is provisionally charged to credit credit card is is is the the the the renter who is responsible responsible the the rental rental .	2.13	Clearly, UNK tokens in the previous case have been removed and more relevant words have been introduced.It is “modality qualifies” that has been more or less captured.

#	Neuron units	seq2seq	Input	Output	Perplexity	Remarks
5b	LSTM	512 units per layer 3 layers 500 iterations	1. A renter may be responsible for more than one advance rental. 2. No open rental may authorize a driver who is barred. 3.Each operating company must have at least one insurer. 4. A rental may be open only if an estimated rental price of the rental is provisionally charged to a credit card that is in the name of the renter who is responsible for the rental. 5.If an estimated rental price of a rental is provisionally charged to a credit card that is in the name of the renter who is responsible for the rental then it is permitted that the rental is open.	1.It is prohibited that who is is for a a driver who who authorized authorized for rental rental . . 2. It is prohibited that the rental of _UNK _UNK _UNK is is for rental rental rental . 3. It is that that the _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK _UNK 4. It is obligatory that an estimated rental price of an open rental is provisionally charged to a credit card that that in name name the the renter who is responsible for for rental . 5. It is obligatory that an estimated rental price of an open rental is provisionally charged to a credit card that that in the name the the renter who is responsible for for rental .	3.35	Since the model is very powerful, more iterations are required

4.4 INFERENCE

PARAMETER TABLE :

The following are the general effects summarised after conducting the above tests by varying the possible hyper-parameters:

Time complexity(number of iterations)	
The greater number of iterations have lesser perplexity with other parameters kept the same . Models with high complexity necessarily need a relatively large number of iterations for producing tangible results. The iterations increased beyond a certain level in a limited corpus run the risk of over-fitting.	
Units used	
The number of units used does not greatly affect the training phase and the testing sentences give higher perplexity even when the number of units used is high.	
Basic cell	
LSTM Being a more complex model when changed in the parameters gives results closer to the intended ones.	GRU The basic gated cell gives more number of UNK (unknown) tokens when used in training

For such a small corpus (of 40 sentences), NMT does not perform well.

NMT models generally need atleast 1000 sentences for reasonable output.

5. CONCLUSION

Natural language processing is gaining precedence and prominence in all walks of life. The primary role of domain experts in the pipeline of translation of natural language to a codified format requires many work hours to be put in. The first step in the process of converting natural language to structured english requires universalisation of the SVBR standards and major acceptance. The use of machine translation is on the rise because of access to high computing power. After skimming the surface of the vast world of MT and the possibilities it holds, we concluded that different approaches have their own merits. The underlying philosophy of every approach has its own merits and demerits. The need is to specifically be able to experiment with newer parameters which cater to the needs of a particular translation process. The requirements from SE to give conclusive output to the next step in overall translation are required to be mapped and categorised.

SMT and NMT approaches have in their own context different tools. NMT being the newer of the two has many upcoming researches in this domain. SMT on the other hand has been around and has now evolved according to the changing requirements. The results found were on a limited data set as the scope of actual data size is larger than the resources of current avail. The comparison of two tools following the mentioned paradigms is purely suggestive in nature and to be used according to the requirements. Finally the proposed future work is based on the challenges faced and the insights gained in the process.

6 . FUTURE SCOPE

1. Both NMT and SMT fail to produce useful results due to restriction of corpora. Linguistic research on the structure of SE can yield in generation of a bilingual corpora.
2. For complex NL structures, like the KYC document, these mechanisms do not produce required results. The KYC document has various constructs and the NL to SE has to be modelled according to the KYC document sense. This restricts the modularity when designing the conversion process.
3. Understanding the nuances of documents requires training a huge corpus on a complex model. Such models could not be executed using the computation power available.
4. Creating a grammar construct to auto generate SE sentences based on an analysed pattern which helps in building the corpus for SE. This needs to have a correspondence in NL for use.
5. Another approach without direct MT is to extract required sentiments and entities from the NL text and auto generate SE sentences. These sentences then can be trained upon a pure English corpus in a neural network so that it learns the right sentence structure and is ready for use.
6. Further standardisation of SE structure to apply to corpus creation.

7. REFERENCES

1. Q. V. Le and T. Mikolov, “Distributed representations of sentences and documents,” arXiv preprint arXiv:1405.4053, 2014.

2. Semantics of Business Vocabulary and Business Rules (SBVR), v1.3 Annex G - EU-Rent Example

3. http://colah.github.io/posts/2015-08-Understanding-LSTMs/

4. http://cs224d.stanford.edu/syllabus.html

5. https://www.tensorflow.org/tutorials/seq2seq

6. https://en.wikipedia.org/wiki/Moses_(machine_translation)

7. https://www.youtube.com/watch?v=beX5rqdneII&list=PLVjXYOjST-AokmIxpCr4GexcdtpeOliBc&index=1

8. http://www.cfilt.iitb.ac.in/Moses-Tutorial.pdf

9. http://www.statmt.org/moses/

10. http://statmt.org/mtma16/uploads/mtma16-neural.pdf

11. https://research.googleblog.com/2016/09/a-neural-network-for-machine.html

12. http://mccormickml.com/2014/06/13/deep-learning-tutorial-softmax-regression/

13. http://ufldl.stanford.edu/tutorial/supervised/SoftmaxRegression/

14. Stolcke, Andreas. "SRILM an Extensible Language Modeling Toolkit." Intl. Conf. on Spoken Language Processing, 2002.

15. A. Lopez. 2008. Statistical machine translation. ACM Computing Surveys, 40(3), Aug. 2008

16. Philipp Koehn , Hieu Hoang , Alexandra Birch , Chris Callison-Burch , Marcello Federico , Nicola Bertoldi , Brooke Cowan , Wade Shen , Christine Moran , Richard Zens , Chris Dyer , Ondřej Bojar , Alexandra Constantin , Evan Herbst, Moses: open source toolkit for statistical machine translation, Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions, June 25-27, 2007, Prague, Czech Republic

17. Bach, N., & Badaskar, S. (2007). A review of relation extraction. Literature review for Language and Statistics II.

18. Sunkle, S., Kholkar, D., & Kulkarni, V. (2016). Comparison and Synergy Between Fact-Orientation and Relation Extraction for Domain Model Generation in Regulatory Compliance. In Conceptual Modelling: 35th International Conference, ER 2016, Gifu, Japan, November 14-17, 2016, Proceedings 35 (pp. 381-395). Springer International Publishing.

19. Sunkle, S., Kholkar, D., & Kulkarni, V. (2016, September). Informed Active Learning to Aid Domain Experts in Modelling Compliance. In Enterprise Distributed Object Computing Conference (EDOC), 2016 IEEE 20th International (pp. 1-10). IEEE.

20. Sunkle, S., Kholkar, D., & Kulkarni, V. Better Regulatory Compliance with Active Participation of Domain Experts.