Phase 3 Report: Semantic Machine Translation (Group 16)

Aditya Prasad Mishra
ASU ID -1211165121
amishr28@asu.edu

Ankit Anand
ASU ID - 1213250712
aanand29@asu.edu

Ashok Prakash
ASU ID - 1214261073
ashokprakash@asu.edu

Laveena Bachani
ASU ID - 1213246721
lbachani@asu.edu

Rohit Bhanushali
ASU ID - 1213179667
rbhanush@asu.edu

1 Introduction

In contrast to the traditional phrased-based statistical machine translation [19] that automatically translates subparts of the sentences, standard Neural Machine Translation (NMT) systems use the sequence to sequence approach at word level and consider the entire input sentence as a unit for translation [20].
The aim of the project is to translate one language to another language using Natural language processing. For this problem, we are using two approaches. In approach one, we use semantic based machine translation with hyperedge replacement grammar. The idea is to generate a Synchronous hyperedge replacement grammar for the source language and assign the weight to its outputs and then convert to target string using target synchronous hyperedge replacement grammar with the best source graph. In another approach, we use Bidirectional LSTM to perform Semantic Machine Translation. Using this approach, we can maintain the essence of the original sentences along with their grammatical structures.

2 Motivation

Machine Translation has huge applications in the commercial, political and military domain. The ability to find information regarding any topic should not be affected because of difference in language speaking capabilities [9]. Furthermore, we may not have sufficient data to map one language to another language that we would like to translate [9]. Our system would employ Semantic Machine Translation. It enables us to maintain the grammatical structure of the languages without disturbing literal meaning of sentences [1], unlike Statistical Machine Translation, which destroys the meaning of the sentence. In this project, we will translate English into Hindi and vice versa [3]. To achieve the stated objectives, we will represent the semantics of a language using a Neural Machine Translation Model. Currently, the available NMT models are implemented as Statistical Models. For Phase One implementation, We studied an existing Semantic Model implementation as described in Section 4 below and created our seq2seq model in Section 5 using a Neural Network Model. In this phase, we explored both ways of machine translation i.e with Language Semantics and Recurrent Neural Network.

3 Related Works

3.1 Semantic Machine Translation Using Knowledge Graph Model

Jones and colleagues et. al, 2012 works on "Semantics-Based Machine Translation with Hyperedge Replacement Grammars", present an approach to semantics-based statistical machine translation that uses synchronous hyperedge replacement grammars to translate into and from graph-shaped intermediate meaning representations [1]. They present algorithms for each step of the semantics-based translation pipeline, including a novel graph-to-word alignment algorithm and two algorithms for synchronous grammar rule extraction.
Our work is influenced by their two alternative rule-based extraction algorithms, one that requires only semantic annotations(utilized using K-Parser) and another that additionally relies on syntactic annotations, the effect of syntax and language bias in meaning representation structures which is effectively the rule-based approach that we are following the

graph generated using the K-parser. The second approach is biased based on the target language since the grammar of the language is specific [1].
In another paper "Towards Translating Natural Language Sentences into ASP", Stefania and Alessio et. al, 2010, build upon recent work by Baral, Dzifcak and Son that defines the translation into ASP of natural language sentences from the lambda-calculus intermediate format generated by CCG grammars.
They introduce an automatic generation of lambda-calculus expressions from template ones, thus improving the effectiveness and generality of the translation process [24]. This approach has been utilized by the graph-based generation logic in K-Parser. Understanding of the knowledge graph helps in correlating the semantic structure of the sentence.

3.2 Machine Translation Using seq2seq Model

Initially, traditional phrase-based translation systems were not that effective because they performed their tasks by breaking the sentences into multiple chunks and then translating those chunks phrase by phrase. This approach led to problems in the translation because this method was completely destroying the grammatical structure and literal meaning of the sentence. The outputs of this approach were not like how humans translate the same sentences. We humans, read the entire document or sentence and understand the complete meaning before translating to another language. NMT uses the similar approach for Machine Translation. Most NMT models focus on how to translate a sentence in a source language to a target language using encoder-decoder neural network [21].
When it comes to NMT, the dominant RNN-based approach is famous. RNNs directly learn the mapping between an input sequence to an output sequence [4]. Whereas Phrase-Based Machine Translation (PBMT) breaks an input sentence into words and phrases to be translated largely independently, Neural Machine Translation (NMT) considers the entire input sentence as a unit for translation. The advantage of this approach is that it requires fewer engineering design choices than previous Phrase-Based translation systems [4]. At first, NMT showed equivalent accuracy with existing Phrase-Based translation systems on processed public benchmark data sets.
RNN-based NMT models perform the translation in a left-to-right manner, leading to the same drawback of underutilization of target-side contexts [22]. To address this issue, Zhang and colleagues [22] first jointly trained both directional LSTM models, and then in testing, they tried to search for target-side translations which are supported by both models.
To handle the defeat of encoding all source-side information into a fixed-length vector, [5] proposed attention-based NMT, which has now become the dominant architecture. However, this model usually suffers from attention failures, which usually lead to undesirable translations [21]. Our Model tries to overcome the drawbacks of all the above systems to give a better Semantic Machine Translation.

4 Example

Let us consider a sentence Anna misses her cat; this can be represented in syntax graph, with parts of speech tagging. Anna(NNP) -> misses(VB) -> her (PRP) -> cat (NN). Then, we can convert the resultant graph into the string using the target language graph and dictionary, which will result in a string " एना अपनी बिल्ली को याद करती हे ."

5 Approach

5.1 Semantic Machine Translation Using Knowledge Graph Model

In this approach, we have worked on the understanding of semantic machine translation through a Knowledge Graph model. The basic underlying concept of the given project is HyperEdge replacement grammars. It converts the English language to knowledge graph using K-parser. Then feed the knowledge graph to target language NLgenerator.

Algorithm:

5.1.1 Preprocessing:

Before generating graph does the preprocessing such as:

- **Name Entity Recognition:** Recognizes the entities in the graph.For example when I gave input "Jonh works for

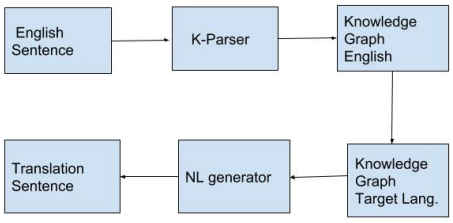

Figure 1: Semantic Translation Flowchart

IBM at Arizona". It returns [IBM = ORGANIZATION, Arizona = LOCATION, Jonh = PERSON].

- **Multiword Expression Finding:** It finds the multiple words in the input and joins them with an underscore. For example NEW YORK: NEW_YORK and Ice cream: Ice_cream.
- **Question-Mark Recognition:** If there exists any question mark. Ir removes that from the sentence before parsing and prints later which output. It also then categorize the sentence into interrogative and searches for related question verbs.

5.1.2 Sentence Graph Generation:

POS Tagging:

The extract method also generates POS tags. POS tags are annotating each word in the sentence with the part of speech tagging, which helps in the lowest level of syntactic analysis and subsequent syntactic parsing and word sense disambiguation.
For our sentence "The Ice cream was eaten by John", generated POS tags are work = VBZ, John = NNP, works - 2 = VBZ, ARIZONA - 6 = NNP, ARIZONA = NNP, act = VBZ, IBM = NNP, person = NNP, etc.

ASP Generation:

After preprocessing it generates the graph for ASP(Answer set representation) to graph parsing node in other language. ASP as Vladimir(2008) states in his paper, "Answer set programming (ASP) is a form of declarative programming oriented towards difficult, primarily NP-hard, search problems. In ASP, search problems are reduced to computing stable models, and answer set solvers(programs for generating stable models) are used to perform the search." [13]

Example: John works for IBM in ARIZONA.

Rules like is_participant_in, is_subclass_of, with_respect_to, in_conjunction_with and prototype_of are figured out in ASP and associated with the words which are annotated from the example text.
The format which generated in the system is given below:

has(IBM-4, is_participant_in, ARIZONA-6).
has(works-2, with_respect_to, IBM-4).
has(works-2, agent, John-1).
has(ARIZONA-6, instance_of, ARIZONA).
has(ARIZONA, is_subclass_of, location).
has(IBM-4, instance_of, IBM).
has(IBM, is_subclass_of, organization).
has(John-1, instance_of, John).
has(John, is_subclass_of, person).
has(works-2, instance_of, work).
has(work, is_subclass_of, act).

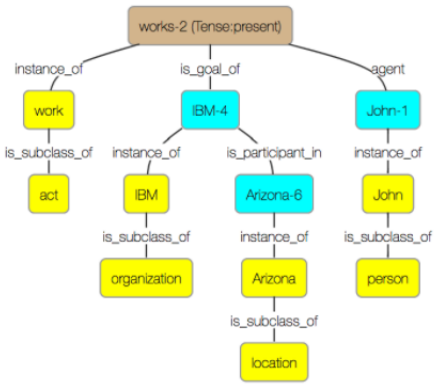

Figure 2: Representation of the knowledge graph for the given example by K-parser

5.1.3 Translation:

After generating English knowledge graph it is converting to target knowledge graph. For that, it is finding the rule between the left and the right annotated words and translating using the translator.
For example, for our given example it generates following ASP. It is using Glosbe API for translation.
has(इब्म् -4, is_participant_in, अरिज़ोन-6).
has(काम करना-2, VBZ, जोह्-1).
has(काम करना-2, with_respect_to, इब्म्-4).
has(काम करना-2, agent, जोह्-1).

5.1.4 NLGeneration:

In NL generation it converts the Target Language to the correct semantic representation of the target text.

- **Semantic Corrections:** The verbs are converted to Target Language into its correct form by appending "hai" and "hain" and "kaam " is converted to "kaam karta hai".
- **Helping verb Conversion:** It checks the English helping verbs and converts them to Hindi. For example, if English word contains "can". In Hindi, it would be converted to "sakta hai".
- **Target Language Generation:** It traverses the ASP rules and generates the language according to the semantic relationship between the words and conceptual class of the words.

Modifications:

Following modifications were made to improve the system's performance.

- **Translation:** Previously, the while translating the words returned by the parser, it used to take by default the first word. Which may not correctly represent the actual meaning of the sentence. For example, not used to translate to "नोट" in Hindi. But now we changed it to "नहीं".
- **Semantic Correction:** Before, there was only support for "they" at some places only. For example there was not correct conversion, in past tense for "they". We improved it to include the translation of "they" to "वे". We also improved the translation of negative past tense sentence. Such as, previously it used to erroneously generate same translation as past participle tense. We corrected it to represent in its correct form.
  Previously while using should and must, it used to keep the same pronoun form in Hindi as helping verb. But actually with Hindi, pronoun changes with the translation. For example, for "He can work on it" translated as "वह इस पर काम कर सकता है". But if we change the helping verb to should it will change from "वह" to "उसे". For example "He should work on it" would translate into "उसे इस पर काम करना चाहिए".
  In English with the change in tense, the verb base form does not change, only the word is modified. For example, "play" changes to played, playing etc. But in Hindi many time the word completely changes with tense, such as "जाना " changes to "गया" or "गई" with change in present to past tense. As we also gave support to plural verb form such as "they" it translates now into "गये" if they are the pronoun.
- **Sentence Structure correction:** We improved the structure of the sentence formation to support negative sentence formation. Previously it used to append the word "नहीं" at random places. We attached it to the verb. For example now, "He should not go there." correctly translated into "उसे उस स्थान पर नहीं जाना चाहिए".

5.2 Machine Translation Using seq2seq Model

5.2.1 Data Preprocessing

For any Machine Learning implementation, Data-Preprocessing plays a vital role to set up the data for the ML algorithm correctly. Our Neural Network needs data in the form of a standard representation which can be utilized for various languages. We chose an integer representation for our model.
Our Corpus Extraction function extracts the data from the source files and generates the separate files for Hindi and English dataset. Once, we have produced the records for English and Hindi separately, We pre-process the generated files and store them in a python array line by line. After this preprocessing step, 'EOS' token is inserted at the end of each line to specify that the sentence ends there.
We then count the words in the python arrays and store this count along with the corresponding word in a dictionary. We consider the words from the word list dictionary, exceeding a certain threshold for training our model. We do that because that would in turn help us to segregate out words used less frequently and Proper Nouns. A token replaces all such words.

We then convert the words to integers by corresponding them to their position in the word list. The index of a word in the word list gives us a one-to-one correspondence. We use this integer representation for both of our natural languages.
We also introduce a token called 'pad' before we put the list of integer arrays for training. Pads fill up the gaps due to varying length of sentences and in turn, provides the convenience to train a batch at a time.

5.2.2 Model

For the development of the underlying Neural Network Architecture, we have used Tensorflow 1.0. We have used an Architecture with 3 Encoders and 3 Decoders with each Encoder and Decoder being an LSTM Cell. The Current section describes the Architecture in detail.

Architecture:

We have employed Bidirectional RNNs (BRNNs) to overcome the shortcoming of conventional RNNs of using only the previous context because, BRNNs process the data in both the directions with two separate hidden layers, which are then feed forwards to the same output layer. [16]
In our model, BRNN computes the forward hidden sequence $\overrightarrow{h}$, the backward hidden sequence $\overleftarrow{h}$ and the output sequence y by iterating the backward layer from t = T to 1, the forward layer from t =1 to T and then updating the output layer [18]:

$$\overrightarrow{h_t} = \mathcal{H}(W_{x\overrightarrow{h}} x_t + W_{\overrightarrow{h}\overrightarrow{h}} \overrightarrow{h}_{t+1} + b_{\overrightarrow{h}})$$
$$\overleftarrow{h_t} = \mathcal{H}(W_{x\overleftarrow{h}} x_t + W_{\overleftarrow{h}\overleftarrow{h}} \overleftarrow{h}_{t+1} + b_{\overleftarrow{h}})$$
$$y_t = W_{\overrightarrow{h}y} \overrightarrow{h}_t + W_{\overleftarrow{h}y} \overleftarrow{h}_t + b_y$$

We are combining BRNNs with LSTM to get bidirectional LSTM [17], which can access long-range context in both input directions. In our model, the input layer is fully connected to the hidden layers, and the last hidden layer is fully connected to the output layer. [18]

Attention Mechanism:

Our project employs Attention Mechanism. This mechanism allows the decoder to attend to different parts of the source sentence at each step of the output generation.

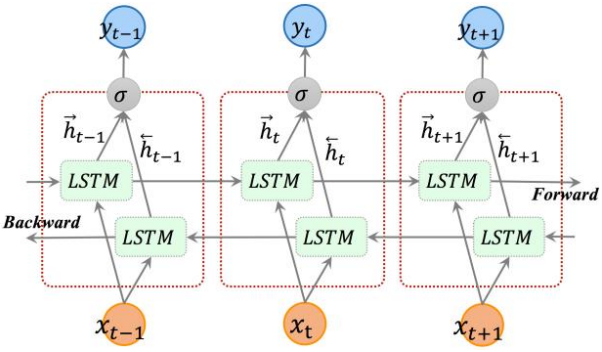

Figure 3: Unfolded architecture of Bidirectional LSTM with three consecutive steps

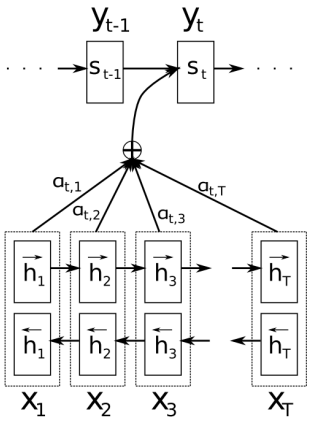

Figure 4: Attention Mechanism

This can be achieved by automatically generating a context vector for each output time step and not a single fixed content vector. Decoder easily generates the output for i'th timestamp by looking into the i'th context vector and the previously hidden outputs. [10]

Calculating i'th Context Vector:

$$e_{ij} = a(s_{i-1}, h_j)$$

'a' is the Alignment model which is a feed-forward neural network that is trained with all the other components of the system.
The Alignment model scores (e) how well each encoded input (h) matches the current output of the decoder (s) [10].
The alignment scores are normalized using a softmax function [10]. The context vector is a weighted sum of the annotations (hj) and

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

normalized alignment scores [10].

Embedding:

We are performing embedding on inputs to map the words to vector of real numbers. The Word-id's were generated from the Word list as described in Data-Preprocessing section. These are embedded into Vectors of specified size. [11] Embeddings are important for training Recurrent Neural Network Models because RNN Models can be trained best on dense vectors, where all values contribute to define an object. The embedding size is a hyper-parameter of our model.

Encoder:

Encoder in our project contains a single Basic LSTM cell. It has Dropout with the probability of 0.5 (50%).
Multi RNN cells are present for encoding the input depending upon the number of layers present.

Decoder:

In our project, Multi RNN cells are there for decoding the input. Dropout with a probability of 0.5 (50%) is present. It contains one Basic LSTM cell.
Decoder returns the training predictions and test predictions after processing data and decoding training and test set. We also define the size of both Encoder and Decoder cells as hyper-parameters. This size is the number of units of LSTM present in a single LSTM Cell.

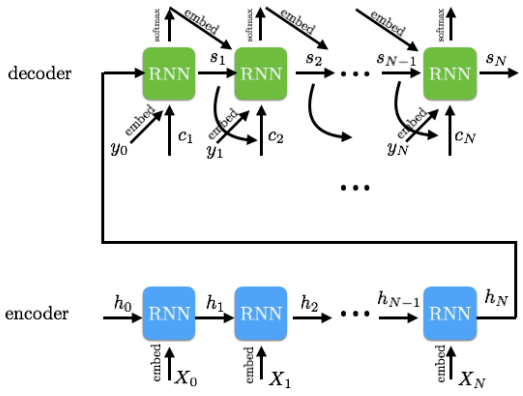

Figure 5: Working of Encoder and Decoder

5.2.3 Training

For initial training purpose, we set the value of epoch to 100, batch size to 64 and learning rate is set to 0.001. We have also set learning decay of 0.9. We have added a dropout equal to 0.5 (50%) in our model to prevent over-fitting of the data. There is a possibility that we will get the highest variance when dropping a neuron with 0.5 probability.
To evaluate our model, we will train the model by changing the hyper-parameters in our model and see if accuracy improves. Our parameter for accuracy is the Best-case batch loss. The batch loss is the total loss of 100 batches. We consider both Training and Validation losses for 100 batches.For better training and validation accuracy, we are doing 10-fold cross-validation on the generated dataset.
Machine Translation Using seq2seq Model

6 Evaluation and Result

6.1 Semantic Machine Translation Using Knowledge Graph Model

Evaluation Parameter:

BLEU:

Bleu is automatic machine translation metric to evaluate the correction of translation when the frequent and quick evaluation is needed. It compares the N-gram candidate with N-gram reference translation and counts the number of matches.
In our algorithm, we took the value of N as 2 for simplicity. We can change it as per our need. N-gram is a sequence of words we are considering to estimating the accuracy. In case of Bi-gram i.e our case, we are checking the pair of words.
Higher the number of N more is the long-term

$$p_n = \frac{\sum_{C \in \{Candidates\}} \sum_{n\text{-}gram \in C} Count_{clip}(n\text{-}gram)}{\sum_{C' \in \{Candidates\}} \sum_{n\text{-}gram' \in C'} Count(n\text{-}gram')}$$

dependency check. We further computed precision by dividing the matches i.e N-gram candidate with N-gram reference translation with the total number of words in the candidate.

Result:

We have tested our model on a small corpus of 100 sentences because 'glosbe API' was blocking the IP above some limit of requests made. Because of the various improvements made to the existing system, we noticed an increment in the 'N-gram Cumulative BLEU' score from **0.23** to **0.31**.

6.2 Machine Translation Using seq2seq Model

Evaluation:

We evaluated the RNN model by training it with a variety of hyper-parameters. The idea was to study the model's accuracy and check if we can improve accuracy by changing some of the hyper-parameters. We observed that the model the performed the best with a learning rate of 0.01 and decay 0.9.The table in Figure 6 documents the results of our experiments with a batch size of 16 with around 5k training examples.

We ran a few more experiments with a larger data corpus of around 50k training examples and a batch size of 128. Graphical analysis of our results is shown in figure 7-10 with corresponding model Hyper-Parameters. With the change in the size of data, we could see the variance in training and validation error decreasing. We could also see, that with the increase in the number of iterations our model gave much higher accuracy. Also from the graphs, it is visible that the validation error actually stabilizes and does not vary by a large margin after a period of training.

Results:

Index for the Table and Graphs:

RNN Size: Number of LSTM units in a single LSTM cell.

Encoding Embedding Size: Size of the embedding matrix for the Encoder.

Decoding Embedding Size: Size of the embedding matrix for the Decoder.

Learning Rate Decay: In every iteration, Learning Rate is decreased by this factor.

Training Loss Error: Average Training Loss Error for 100 batches.

Validation Loss Error: Best average Validation Loss Error for 100 batches.

RNN Size	Encoding Embedding Size	Decoding Embedding Size	Learning Rate	Learning Rate Decay	Results	
					Training Loss Error *	Validation Loss Error*
64	256	256	0.01	0.9	0.810	2.26
256	256	256	0.01	0.9	0.804	2.314
512	256	256	0.01	0.9	0.873	2.356
64	512	512	0.01	0.9	0.787	2.334
64	512	512	0.07	0.9	0.795	2.363
64	512	512	0.05	0.9	0.866	2.306
64	256	256	0.07	0.7	1.105	2.836
32	256	256	0.01	0.95	0.901	2.311

Figure 6: Observations with smaller corpus

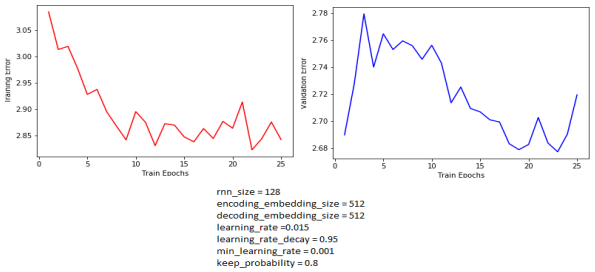

Figure 7: Model 1

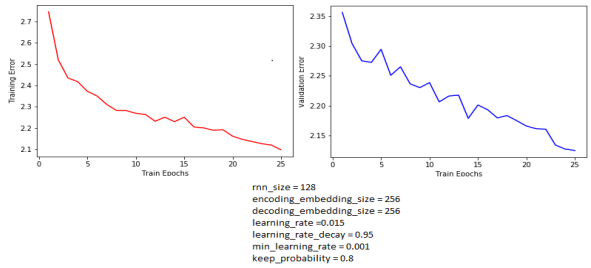

Figure 8: Model 2

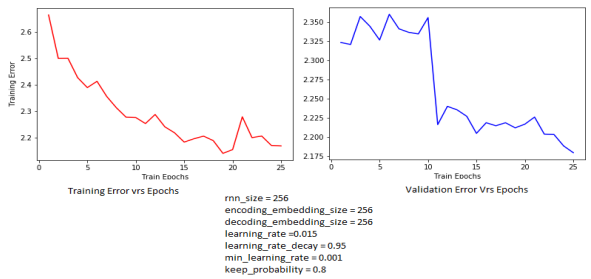

Figure 9: Model 3

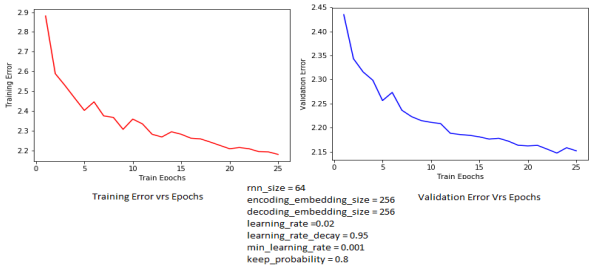

Figure 10: Model 4

7 Error Analysis

7.1 Semantic Machine Translation Using Knowledge Graph Model

Translation system tries to address most of the grammatical correctness in the language. However few prepositions are not handled in the system, which doesn't affect the understanding of the sentence, but syntactically the sentence structure is wrong. For example, John works at IBM in Arizona, is translated to, जॉन एरिज़ोना में आईबीएम पर काम करता है. Here "जॉन एरिज़ोना में आईबीएम पर काम करता है" should be "जॉन एरिज़ोना में आईबीएम के लिए काम करता है".

This sentence gives a good understanding of the translated context, but this is syntactically not correct in the language. As per our study on the parser, the K-Parser is able to parse this input and it is able to produce proper semantic representation. The transliteration fails to change the source to target language properly which is why the discrepancy exists in the output. Multiple prepositions like this were also analyzed and the improvements in the transliteration as per the context is still required.
Some sentences in the Kparser are not handled correctly, which resulted in exception. For example "Who are you ?" It does recognize question in this sentence.

The translation of the literal words is handled by taking directly the first word returned by the dictionary. It does not handle the word sense disambiguation or appropriate word for the context of the sentence given. We can propose various techniques for such translation.

Questions other than "Who" and "How" are not handled. For example binary questions such as "Would you like to go out?" or "Do you like this?" are handles by the kparser but it throws error while putting into the system. This is because the tree structure of such questions is not considered while translating to another language.

7.2 Machine Translation Using seq2seq Model

Analysis of the error rate gave us a good insight into our model. We are mentioning a few factors down below which impacted the accuracy of our model.
Hyper-parameters - Any given Neural Network Model requires fine tuned Hyper-parameters for training. Hyper-parameters play a very important role and many times it is difficult to deduce them only by looking at the model or the data. Hyper-parameter search could be a major boosting factor for improving accuracy. Even though we tried our best to fine tune various Hyper-parameters, selected Hyper-Parameters could have impacted the accuracy.

Amount Of Data - The model can get only better by training it with a large amount of varying data. Also, the variance in any given corpus is a very important factor. We could see the divergence in training and validation error decreasing with increase in examples. We could not identify a corpus with higher variance which could have impacted the accuracy.

Miscellaneous - We also faced many Miscellaneous challenges. Some of these include Data clean up issues and issued due to a Shared GPU. Data Clean up issues was difficult to spot because certain characters which needed cleanup could be spotted by observing the corpus file very carefully. One good example is ‘ is different from ’. Spotting of all such issues is almost impossible from the given corpus. Even after careful examination some of these might have crept into the corpus and could have impacted our accuracy. Usage of a Shared GPU did hold us back because we could not train for the amount of time we wanted as the Server would exhaust any process running for more than a certain amount of time. We could only train for at max for 4-5 hours. Due to multiple users using the GPU at the same time we were only given a piece

of the memory, so we have to accordingly plan our experiments else we would receive Out of Memory errors. Both of these factors could have impacted our overall accuracy.

8 Conclusion and Future Work

In the first approach of our project, we proposed to improve the accuracy of the translation from the existing method proposed in hyperedge replacement grammars. Our analysis on parser output led us to handle the replacements of the language at the knowledge graph level, which showed considerable results in the translation. Results also show that our question handling in few cases outperforms the previous project.

Right now we haven't tried semantic extractions with appropriate syntax in the sentence. Works on sentence context extraction by lexical analysis, identifying root words, grammatical information and semantic categories can result spotting the suitable prepositions or transliterations to the sentence semantics. Also, one more approach which is proposed for English to Bengali in [25] for identifying the morphological characteristics of the prepositions can also help in solving the issue for Hindi translation. Identifying verb and prepositions separation is also required, for example, *held on* is represented as *heldon* together which accounts as a verb and not as a preposition. These can also be solved to improve the translation accuracy.

In the second approach, we have equipped the conventional RNN with Bidirectional LSTM, encoder-decoder and attention mechanism. With changes in the hyperparameters of our model, we have observed the training loss error, validation loss error and accuracy of our model. Experimental results on English-Hindi translation tasks demonstrate the effectiveness of our model.

In our Neural Network model, Gated recurrent units (GRUs) and Variational Autoencoders (VAEs) with minor changes in the hyperparameters can be used for better results. By using the state of the art dataset, the current Bidirectional LSTM model can be tested and the corresponding results can be compared to the results of the other models mentioned in related works.

References

[1] Jones, Bevan, et al. "Semantics-Based Machine Translation with Hyperedge Replacement Grammars." COLING. 2012

[2] Chen, Yun, Yang Liu, Yong Cheng, and Victor OK Li. "A Teacher-Student Framework for Zero-Resource Neural Machine Translation." arXiv preprint arXiv:1705.00753

[3] Ekansh Gupta, Rohit Gupta, et al. "Hindi English Parallel Corpus Generation from Comparable Corpora for Neural Machine Translation" CSE - IIT Kanpur

[4] Ilya Sutskever, Oriol Vinyals, Quoc V. Le, et al. "Sequence to Sequence Learning with Neural Networks" Conference on Neural Information Processing Systems

[5] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, et al. "On the Properties of Neural Machine Translation: Encoder–Decoder Approaches" arXiv:1409.1259

[6] Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, et al. "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation" arXiv:1406.1078

[7] Minh-Thang Luong, Quoc V. Le, Oriol Vinyals, et al. "Multi-Task Sequence to Sequence Learning" ICLR 2016

[8] Dzmitry Bahdanau, KyungHyun Cho, Yoshua Bengi, et al. "Neural Machine Translation by jointly learning to Align and Translate" ICLR 2015

[9] Miles Osborne, et al. "Statistical Machine Translation" University of Edinburgh

[10] Pranoy Radhakrishnan, et al. "Attention Mechanism in Neural Network"

[11] Thang Luong, Eugene Brevdo, Rui Zhao, et al. "Neural Machine Translation (seq2seq) Tutorial"

[12] Piotr Chabierski, et al. "Logic-based

Approach to Machine Comprehension of Text" Student-projects-ug-2016-17

[13] Vladimir Lifschitz, et al. "What Is Answer Set Programming?" University of Texas at Austin

[14] "Deep Learning and NLP A-Z" Course on SuperDataScience Website

[15] Kishore Papineni, Salim Roukos, Todd Ward, Wei-Jing Zhu, et al. "BLEU: a Method for Automatic Evaluation of Machine Translation" ACL, July 2002, pp. 311-318

[16] Mike Schuster, Kuldip K Paliwal, et al. "Bidirectional recurrent neural networks" Signal Processing, IEEE Transactions on, vol. 45, no. 11, pp. 2673–2681, 1997

[17] Alex Graves, Jurgen Schmidhuber, et al. "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," Neural Networks, vol. 18, no. 5, pp. 602–610, 2005.

[18] Zhou Yu, Vikram Ramanarayanan, David Suendermann-Oeft, Xinhao Wang, Klaus Zechner, Lei Chen, Jidong Tao, Aliaksei Ivanou, Yao Qian†, et al. "Using Bidirectional LSTM Recurrent Neural Network to learn high-level abstractions of sequential features for Automated scoring of Non-Naive Spontaneous Speech" Educational Testing Service R&D

[19] Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., Herbst, E., et al. "Moses: open source toolkit for statistical machine translation" ACL 2007, pp. 177–180. Association for Computational Linguistics, Stroudsburg (2007)

[20] Mercedes Garcia-Martınez(B), Loic Barrault, Fethi Bougares, et al. "Neural Machine Translation by Generating Multiple Linguistic Factors" SLSP 2017, LNAI 10583, pp. 21–31, 2017

[21] Jinsong Su, Shan Wu, Deyi Xiong, Yaojie Lu, Xianpei Han, Biao Zhang, et al. "Variational Recurrent Neural Machine Translation" arXiv:1801.05119

[22] Xiangwen Zhang, Jinsong Su, Yue Qin, Yang Liu, Rongrong Ji, Hongji Wang "Asynchronous Bidirectional Decoding for Neural Machine Translation" arXiv:1801.05122

[23] Rico Sennrich, Barry Haddow, Alexandra Birch, et al. "Neural Machine Translation of Rare Words with Subword Units" Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, 2016

[24] Stefania Costantini, Alessio Paolucci, et al. "Towards Translating Natural Language Sentences into ASP" Universita di L'Aquila, Italy

[25] Sudip Kumar Naskar, Sivaji Bandyopadhyay, et al. "Handling of Prepositions in English to Bengali Machine Translation" Dept. of Comp. Sc. and Engg., Jadavpur University, ResearchGate