WebNLG Phase 2 Class Project

By
Saptarashmi Bandyopadhyay
Mitchel Myers
Namita Kalady Prathap
Yuchen Sun

Department of Computer Science and Engineering

Pennsylvania State University, University Park

Class: CSE 597, Section 1, Natural Language Processing

Submission Date: April 21, 2019

Contents

1. Introduction

Natural Language Generation refers to the process in which structured data is converted into a human-readable format such as a sentence or paragraph. RDF is a structured format for describing data in the form of (subject, predicate, object) triples (hereafter referred to as triples). RDF is often associated with the Semantic Web; which is based on the idea that all data on the Internet should be stored in a uniform and computer-accessible manner, namely the RDF format, in our Project. DBpedia is one such effort at attaining a Semantic Web. DBpedia stores a large RDF graph which is accessible over the Internet. This final project report describes an attempt at the WebNLG challenge.

The WebNLG challenge focuses on the microplanning sub-task of natural language generation (NLG). The goal is to take as input content represented as RDF triples and output an English lexicalization (or verbalization) of the content. This task is important because the way in which the data is stored in RDF as a graph is not the most human-accessible representation of the data. Having an automated means for generating makes the data more usable in general (e.g. as the last step in DBpedia query tool).

This report is a phase 2 approach following earlier attempts. After analysing the baseline system in phase 1 and identifying its weaknesses, we applied our changes to the baseline system. It has been observed that the BLEU scores of our proposed system generating delexicalised output is comparable to the scores of the systems from the WebNLG 2017 challenge considered for comparison after relexicalisation.

The pre-processing and post-processing steps have been implemented in our system as follows. At first, the dataset is split into the training, development and testing datasets. Then, delexicalisation is used during our pre-processing step by replacing named entities with DBpedia categories, to allow for generalization. Other pre-processing steps like byte pair encoding for vocabulary files and joining multiple tokens in a predicate or object have been applied on the datasets. Then, the baseline NMT model, using the Tensorflow NMT system, is trained on the delexicalized data and the NMT translation model is obtained. This translation model is used to translate the test triple sets and produce delexicalised output lexicalisations. Finally, in post-processing, the delexicalised outputs are relexicalised to produce the final lexicalizations. The DBpedia categories are replaced by the tokenized named entities during relexicalisation. BLEU scores are calculated and compared with the WebNLG 2017 Baseline system as well as with other systems that participated in WebNLG 2017 with NMT systems. It has been observed that the BLEU scores obtained after relexicalisation is better than most of the systems in the WebNLG challenge.

2. Data Preparation

The provided training and development files are split into the training, development and testing datasets as described in Section 3. These files are henceforth referred as the 20k dataset due to the total number of lines in the training, development and testing dataset being 20283. Due to paucity of data, the unseen test data is considered along with the provided training and development files, which are then split into the training, development and testing datasets as described in Section 3. These files are henceforth referred as the 22k dataset due to the total number of lines in the training, development and testing dataset being 22716. The 20k and 22k datasets have different delexicalisation dictionaries in our Project as all the entities in the subject of the triples in the 2 datasets are mapped to their DBpedia categories. The testing of the unseen dataset is also done separately by using the training, and development files of the 20k dataset and by delexing the unseen test file with the delex dictionary of the 20k dataset. The statistics of the 20k and the 22k dataset have been presented in Table 1.

Dataset	**Size of Training file (in number of lines)**	**Size of Development file (in number of lines)**	**Size of Testing file (in number of lines)**	**Total number of lines in the training, development and testing files**
20k Dataset	16226	2029	2028	20283
22k Dataset	18172	2269	2275	22716

Table 1. Dataset Statistics

The dataset consists of one or more RDF triples (a triple set) on the source side and one or more lexicalizations on the target side. The (triple, lex) pair files are organized on the basis of the number of RDF triples and within each file on the basis of the DBPedia category of the subject of the triple.

The first level of pre-processing carried out can be categorized as:
• Separation of the subject, predicate and object by a blank space replacing the pipeline symbol.
• Linearization of multiple triples for a single entry in the XML files.
• Repetition of the linear set of RDF triples for every lex in each entry in the XML files.

The delexicalisation (Gardent et. al., 2017) of the source triples and the target lex forms was carried out after the initial pre-processing in Phase 1 of the project.. The delex_dict.json file provided by WebNLG was used in Phase 1. It contains a list of entities mapped to 10 seen Dbpedia categories in the data and 5 unseen categories. More entities were added in the *seen* and *unseen* categories. A new *unseen* category COUNTRY has been added. For the RDF triple <s,p,o> where s refers to subject, p refers to the predicate and o refers to the object.
• s is replaced by the DBPedia category in upper case.
• o is replaced by p in upper case

The process of delexicalizing both the triple and lex forms in a training pair generalizes the generation of the lexicalized form from a triple set. This helps in modeling the triple to lex language generation problem as a machine translation task.

Delexicalisation is useful to capture the semantic representation of RDF and lex. Thus, the replacement of s and o are replicated on the lex side as well. e.g.
• RDF triple before delexicalisation : - *Lahore Allama Iqbal International Airport runwayName 18L / 36R*
• RDF triple after delexicalisation : - *AIRPORT runwayName RUNWAYNAME*

The subject in the triples and the lex file was replaced by its corresponding category, only after direct matching before tokenization. This was useful to correct some mistakes in the WebNLG pre-processing where some subjects with entries in delex_dict had not been replaced. Similar logic had been applied to replacing the object by the predicate. The string matching in the proposed baseline was case-insensitive. Thus, the following erroneous triple in WebNLG baseline “A . S . Roma league LEAGUE A . S . Roma ground GROUND” was corrected to
“SPORTSTEAM league LEAGUE SPORTSTEAM ground GROUND”

It had been observed in the WebNLG implementation that, erroneously, the subject was being updated with categories based on the object (replaced by predicate) in preceding triple which was not mentioned in the paper. It could have inflated the BLEU scores in the WebNLG paper. This had been corrected in the Phase 1 of the project.

An erroneous triple in WebNLG baseline is highlighted where the country United States is getting replaced by the predicate in the preceding category although country is not present in the delex_dict.json provided by WebNLG:

- WRITTENWORK country COUNTRY COUNTRY ethnicGroup Native Americans in the COUNTRY

The baseline model in Phase 1 corrected this error by retaining United States as subject and correctly replacing the object by the predicate.

- WRITTENWORK country COUNTRY COUNTRY ethnicGroup ETHNICGROUP

The processing on the data in the Phase 2 of the project has been improved over the baseline model in Phase 1. The delexicalisation process has been updated and other pre-processing steps have been incorporated. Relexicalisation has also been executed. These improvements in the data processing are discussed in the following sections.

2.1 Improved Delexicalisation

The delexicalisation process has been updated to ensure that the DBpedia categories for all the named entities in the train, dev and test data for the 20k dataset and 22k dataset are extracted by querying the DBpedia knowledge base, using SPARQL. SPARQL is a RDF query language which is able to retrieve and manipulate data stored in the RDF format. All entities were accessible as DBpedia resources (prefixed by “http://dbpedia.org/resource/”). Furthermore, all entities are part of DBpedia’s rich ontology system, which provides type information about resource entities through the RDF predicate “rdf:type”. Each named entity may belong to many RDF types. The process for assigning one category for delexicalisation is as follows:

1. If an entity belongs to a type which matches the name of the category of the file it appears in, then that category is automatically assigned to that entity. This corresponds to how the original test data was prepared, i.e. the triples in the dataset for a particular category were obtained by starting from root entities which have the category as a type.
   1.1. E.g. Aarhus_Airport is a subject in the 1triple_allSolutions_Airport_train_challenge.xml file in the 1 triples folder of the provided train folder. When the category of the entity is being queried by SPARQL from DBpedia, 23 types are generated of which one of them is “Airport”, which is present in the name of the file. So, the entry for mapping this entity to a category in the delexicalisation dictionary is “Airport”:”Aarhus_Airport”.
2. For entities whose type could not be automatically obtained by satisfying the condition 1 as mentioned above, a manual selection was made from the types available for the category. Using DBpedia’s ontology system’s hierarchy of types, we were able to see which types were at the lowest points of the hierarchy (more specific types) and preference was given to these types, as they provided more information.
   2.1. E.g. Spain is present as a subject in triples in 1triple_allSolutions_Food_train_challenge.xml in the 1 triples folder of the provided train folder. Now, the 29 types, extracted from DBpedia, do not contain “Food” which is present in the name of the file. Then, by manually observing the context in which the triples are present, the named entity, in this case “Spain”, is assigned a category from the 29 types, which is “Country” in this case. So, the entry for mapping this entity to a category in the delexicalisation dictionary is “Country”:”Spain”.

This provided an improvement over the base model, in which such entities were not delexicalised because they did not have an entry in the delexicalisation dictionary. There were only 16 categories in the delexicalisation dictionary, in Phase 1 of the project. However, in Phase 2, there are 43 categories for the 20k dataset and 58 categories for the

22k dataset, which has been illustrated in Table 2. Our delexicalisation process ensured that all the entities present in the subject position were delexicalized correctly, allowing for better generalization.

	20K Dataset	**22K Dataset**
Total DBpedia Categories	43	58
Total number of Entities	434	719
Entities Assigned Automatically	208	323
Entities Assigned Manually	226	396

Table 2. Statistics of the Updated Delexicalisation Dictionary

The delexicalisation of the object was similar to that of the baseline, in which the object was replaced by the predicate in uppercase (e.g. for the “location” predicate, the object was delexicalized as “LOCATION”).

2.2 Modification of the Vocabulary for Training

The Vocabulary has been constructed in 2 ways. One method is by adopting the same procedure as in Phase 1 of the experiment, considering each unique word in the triple and lex files for the train and the development datasets as each line in the Vocabulary file for triples and lex. The statistics of the Vocabulary file are presented below in Table 3. It can be observed, that the size of the vocabulary file reduces after delexicalisation which makes the dataset more general and helps in faster training.

Another mechanism of creating the vocabulary files is by Byte Pair Encoding (BPE), to generalize the vocabulary for both the triples and lexicalisations. BPE is an approach that segments the corpus in such a way that frequently occurring sequence of characters are combined. It results in having word surface forms divided into its root word and affix. It alone handles out-of-vocabulary words, but tends to not consistently segment inflected words. The Subword NMT tool [5] has been used for the unsupervised word segmentation. The subword units are generated and then the shared BPE is learned to generate the vocabulary file for BPE. BPE vocab improves the vocabulary and hence the BLEU score.

The statistics of the vocabulary files are presented below in Table 3.

Vocabulary Size	**20K dataset**	**22K dataset**
Source (triples) before delex	4931	5964
Source (triples) after delex	527	750
Target (lex) before delex	5908	7603
Target (lex) after delex	2449	3629
Using BPE (triple + lex)	4677	5722

Table 3. Vocabulary Statistics

2.3 Multiple Reference Files for BLEU score calculation

It has been noted during the pre-processing of the data that one chain of triples can have multiple lexicalisations in the data. Thus the same chain of triples are repeated for every possible lexicalisations, leading to repetitions of the triple chains in the triple files for training, development and testing. The size (number of lines) of the test files and the reference files for the 20k dataset, 22k dataset and the unknown dataset have been presented in Table 4.

Therefore, multiple reference files are created from the lex files. The number of reference files is the maximum number of lexicalisations, a chain of triples can have in the dataset. Each reference has a lexicalisation, for a particular triple. Each reference file has index number which indicates the possible lexicalisation in the dataset. E.g. test-referene0.lex file indicates that it has the first lexicalisation for the unique triples in the dataset. As, the index number increases, the reference files become sparse due to the fact that not all triples have a large number of lexicalisations. This allows the BLEU score not to be skewed by the number of times a triple chain appears in the test input. The BLEU scores of the generated lexicalisation improves significantly on using the multiple reference files.

	Size (Number of Lines) of the test files	**Size (Number of Lines) of the reference files**
Test Data for 20k Dataset	2028	780
Test Data for 22k Dataset	2275	862
Unseen Test Data	2433	891

Table 4. Size of the Reference Files and the Test Files

2.4 Improved Pre-Processing of the Delexicalised Triples

The WebNLG baseline model, which we had adopted in Phase 1 of the Project, generated predicates and objects separated by blank spaces ‘ ‘ in between multiple tokens after delexicalisation.
e.g. A triple was “ASTRONAUT was selected by NASA WAS SELECTED BY NASA”

However, the subject, predicate and object are also joined by a blank space. So, it is difficult to distinguish the subject, predicate and the object in the triples. The problem becomes aggravated for chains of triples where each triple is separated by blank spaces. To solve this problem, the pre-processing has been updated in such a way, that multiple tokens in the predicate and object after delexicalisation are joined by ‘_’. It helps in differentiating, the tokens in the predicate and the object with the subject, predicate and object in the triple.
E.g. The above triple can be joined to “ASTRONAUT was_selected_by_NASA WAS_SELECTED_BY_NASA”

2.5 Relexicalisation

A prominent post-processing step has been implemented in Phase 2 of the project which is relexicalisation of the delexicalised system generated lexicalisation files. The relexicalisation process can be defined as the process of converting the categories in the delexicalized file to corresponding tokenized entities for the category in the source triple file. Relexicalisation is done to obtain the BLEU score by considering the source lexicalisation file, before it was subjected to delexicalisation. The relexicalisation is done on the category, because for each lexicalisation, the category may represent multiple entities, one of which is the entity obtained from the corresponding source triples.

Relexicalisation was achieved by maintaining a relex dictionary file, created for each chain of triples while the data is being delexicalised. Each triple chain has a separate dictionary which contains the mapping of the named entity to the category. The dictionary for each triple chain is maintained in a line in the relex dictionary corresponding to the line of the chain of triples. The system generated delexicalised file is provided as input to the relexicaliser which refers to the particular dictionary in the file for each lexicalisation; the line number of the dictionary in the relex dictionary file being the same as the line number of the lexicalisation in the delexicalized file. The extracted entity is then tokenized by replacing the underscores with blank spaces. A separate issue was the relexicalisation of objects with quote characters to signify data values or particular objects. The quote characters were removed and then the tokens in the entity were obtained by running the tokenizer of NLTK 3.0, toolkit [6].

An example of delexicalisation and subsequent relexicalisation can be presented as follows:

Initial triple: Taylor_County,_Texas largestCity Abilene,_Texas
Initial lex: Abilene is the largest city in Taylor County , Texas .
Delexicalised triple: LOCATION largestCity LARGESTCITY
Delexicalised lex: Abilene is the largest city in LOCATION .
Dictionary entry in the Relex dictionary file: [["Taylor_County,_Texas", "LOCATION"], ["Abilene,_Texas", "LARGESTCITY"]]
Output lex after testing: The nearest city to LOCATION is NEARESTCITY .
Relexicalised lex: The nearest city to Taylor County, Texas is NEARESTCITY .

It should be noted, that “Taylor County , Texas” is delexicalised to LOCATION category in the lex and relexicalised back to “Taylor County , Texas “ from the generated lexicalisaton after testing. However, it should also be observed that “Abilene,_Texas” did not get delexicalised as the tokenized entity cannot be identified by direct matching with the initial lex. Consequently, “Abilene” is not relexicalised in the generated lexicalisation.

3. Train/Test Split

The provided dataset is split into the training, development and testing datasets by adopting the unconstrained splitting mechanism for the WebNLG challenge dataset (Shimorina and Gardent, 2018). This mechanism ensures that the original dataset is split into the train/dev/test datasets following a proportion of 80:10:10 such that no chain of triples are common in the three datasets. The splitting of the three datasets is done randomly, to remove any inherent bias in the train, dev and test datasets.

For every file in the provided training and the development folders with the same category, the total triple and lex entries in each XML file for different sizes of triples are distributed among the train, dev and test sets in 80:10:10 proportion. This ensures that the proportional distribution of the data in terms of the different DBpedia categories and the size of triples are maintained in the three types of datasets. It has also been ensured that there is no overlap among the training, development and test datasets before delexicalisation, i.e., for each distinct triple set (after delexicalisation), all the lexicalizations belong to only one of the datasets. For each triple in the triples file, its corresponding lexicalisation is present in the same line number in the lex file. The triples and lexicalisations are randomly shuffled by using the same seed value for the triple and lex to ensure, that they are aligned in the triple and lex files, even after randomization.

4. Model and Training Parameters

The Tensorflow NMT tool expects pairs of files representing input translations. A line in the first file translates to the corresponding line in the second file. The first file is the input language (RDF triples) and the second file is the output language (English lexicalizations). Each line should be a spaced-delimited sequence of tokens. A pair was needed for each of the training (train), parameter-tuning (dev), and testing (test) sets. Additionally, the NMT library needs vocabulary files where each line represents one word/token in the language. The first three tokens needed to be '<s>', '</s>', and '<unk'> for start-of-sentence, end-of-sentence, and unknown-tokens. The following training parameters have been adopted for the Project based on the (Shimorina and Gardent, 2018) paper

- 2 layered uni-directional recurrent neural network with LSTM (long short term memory) as the recurrent unit whereas WebNLG had used a bidirectional encoder-decoder architecture. Experiments have also been conducted with a 4 layered uni-directional recurrent neural network, to study any improvement in the BLEU score by increasing the layers.
- 500 hidden layers (as per WebNLG)
- 0.3 Drop-out rate (as per WebNLG) and 0.5 Drop-out rate, to study the impact of increasing the drop-out rate.
- Batch size = 128
- Learning rate: The initial learning rate is 1.
- Attention models: standard (as per WebNLG), scaled_luong and normed_bahdanau models have been used as described below.

We have also made a successful attempt to figure out the influence of attention models on the delexicalised data. The attention models connects different parts of the source and target with different weights, which helps distinguish keywords in sentences and benefits the translation of long sentences. The tensorflow NMT model provides four kinds of attention models, in addition to the standard attention model. In the Bahdanau attention model, a context vector is derived from the hidden states of a source in the previous timestamp in order to predict the current target. Luong's technique improves the Bahdanau attention model by creating an independent RNN-like structure to take the concatenation of context vector and the current hidden state [3]. Moreover, it simplifies Bahdanau's technique by using hidden states at the top LSTM layers in both the encoder and decoder, It also simplifies alignment functions, adding dot function that works better than concat function in global attentional model, and general function that works well in local attentional model. This modification decreases the computation costs of the attention mechanism.

The "normed_bahdanau" and "scaled_luong" model are the Bahdanau and Luong model with weights normalization based on Salimans and Kingma's paper [4]. According to Salimans and Kingma, while the architectures of neural networks differ widely across applications, they are typically mostly composed of conceptually simple computational building blocks sometimes called neurons: each such neuron computes a weighted sum over its inputs and adds a bias term, followed by the application of an element-wise nonlinear transformation. Indeed, improving building blocks improves the performance of a very wide range of model architectures and could thus be very useful. They re-parameterize the weight vectors in a neural network that decouples the length of those weight vectors from their direction, and prove that this technique improves the conditioning of the optimization problem and speeds up convergence of stochastic gradient descent. After researching these models, we chose "normed_bahdanau" and "scaled_luong" attention model for tests. We tested them on four trials, and we have noticed that the "scaled_luong" attention model outperforms the "normed_bahdanau" mechanism on average, so we decide to use "scaled_luong" as our attention model in later experiments.

5. Results

The results of our experiments are presented in tables 5, 6 and 7. We display the influence of number of layers, drop out rate, attention models, delexicalisation, relexicalisation, unseen data, and vocab file on the BLEU scores.

The BLEU (bilingual evaluation understudy) metric is used to evaluate the quality of a text, generated by a machine translation system from one natural language text to another. Quality can defined as the closeness between the output of a machine translation system to that of a human translated output. BLEU scores are calculated for individual translated segments, i.e., sentences, by comparing them with a set of good quality reference translations. The average of such scores over the whole corpus provides an estimate of the overall quality of the translation. However, BLEU score does not take into account the coherence or grammatical correctness of the output. The value of the BLEU score is a number between 0 and 100. A BLEU Score nearer to 100 indicates that the candidate translation is closer to the reference translation while a score nearer to 0 signifies that the candidate translation is far away from the reference translation. BLEU scores are calculated by counting the number of matching n grams in the candidate translation to n grams in the reference text. The modified n gram precision techniques for counting of matching n grams takes the occurrence of the words into the reference text into account and not rewarding a candidate translation that generates an abundance of reasonable words.

We will discuss each of the experiments in details in the next section. Table 5 presents the BLEU scores for the normal and BPE vocab files with the Standard attention model for the data (Seen Categories - 20k and All Categories - 22k). Similarly tables 6 and 7 present the results for these vocabulary files for the Scaled Luong and Normed Bahdanau attention models respectively. The columns “Dropout”, “Encoder Type”, “Attention model”, “Number of Layers” in the tables 5, 6 and 7 show the influence of the drop out rates (0.3 and 0.5), type of encoding selected, attention model (Standard, Scaled Luong and Normed Bahdanau) and the number of layers selected on the BLEU scores for the delexicalisation and relexicalisation processes respectively.

Dropout	**Encoder Type**	**Attention model**	**Number of Layers**	**BLEU Scores [delex]**	**BLEU Scores [relex]**	**Data**	**Vocab file**
0.3	Unidirectional	Standard attention model	2	dev bleu = 19.7 test bleu = 18.1	dev bleu = 38.83 test bleu = 34.68	20k	normal
0.5	Unidirectional	Standard attention model	2	dev bleu = 19.3, test bleu = 19.1	dev bleu = 39.87 test bleu = 39.03	20k	normal
0.3	Unidirectional	Standard attention model	2	dev bleu = 19 test bleu = 18.5	dev bleu = 39.10 test bleu = 36.48	20k	BPE
0.5	Unidirectional	Standard attention model	2	dev bleu = 18.6 test bleu = 18.4	dev bleu = 40.20 test bleu = 40.93	20k	BPE
0.3	Unidirectional	Standard attention model	2	dev bleu = 19 test bleu = 18.5	dev bleu = 37.10 test bleu = 36.48	20k	BPE

0.3	Unidirectional	Standard attention model	2	dev bleu = 18.0 test bleu = 18.7	dev bleu = 40.00 test bleu = 35.81	22k	normal
0.5	Unidirectional	Standard attention model	2	dev bleu = 18.3 test bleu = 19.7	dev bleu = 39.74 test bleu = 37.25	22k	normal

Table 5. Results - BLEU scores for normal and BPE vocab files with the Standard attention model.

Table 6 presents the results for the normal and BPE vocabulary files for the Scaled Luong attention model below. The process of creating vocabulary files in the traditional way and by BPE has been described in Section 2.2.

Dropout	**Encoder Type**	**Attention model**	**Number of Layers**	**BLEU Scores [delex]**	**BLEU Scores [relex]**	**Data**	**Vocab file**
0.3	Unidirectional	Scaled Luong	2	dev bleu = 23 test bleu = 21.5	dev bleu = 45.05 test bleu = 40.07	20k	normal
0.5	Unidirectional	Scaled Luong	2	dev bleu = 24.1 test bleu = 21.7	dev bleu = 44.43 test bleu = 40.47	20k	normal
0.3	Unidirectional	Scaled Luong	2	dev bleu = 24.5 test bleu = 22	dev bleu = 46.33 test bleu = 41.33	20k	BPE
0.5	Unidirectional	Scaled Luong	2	dev bleu = 25.5 test bleu = 23.7	dev bleu = 45.28 test bleu = 41.51	20k	BPE

Table 6. Results - BLEU scores for normal and BPE vocab files with the Scaled Luong attention model.

As seen below, table 7 presents the results for the normal and BPE vocabulary files for the Normed Bahdanau attention model. The process of creating vocabulary files in the traditional method and by BPE has been described in Section 2.2.

Dropout	**Encoder Type**	**Attention model**	**Number of Layers**	**BLEU Scores [delex]**	**BLEU Scores [relex]**	**Data**	**Vocab file**
0.3	Unidirectional	Normed Bahdanau	2	dev bleu = 22.6 test bleu = 20.8	dev bleu =40.2 test bleu = 39.8	20k	normal
0.5	Unidirectional	Normed Bahdanau	2	dev bleu = 23.2 test bleu = 21.8	dev bleu =42.1 test bleu =39.7	20k	normal
0.3	Unidirectional	Normed Bahdanau	2	dev bleu = 20.9 test bleu = 19.7	dev bleu = 41.31 test bleu = 40.59	20k	BPE
0.5	Unidirectional	Normed Bahdanau	2	dev bleu = 22 test bleu = 21.2	dev bleu = 43.83 test bleu = 42.67	20k	BPE
0.5	Unidirectional	Normed Bahdanau	2	dev bleu = 23.2 test bleu = 22.8	dev bleu = 44.15 test bleu = 41.5	22k	normal

Table 7. Results - BLEU scores for normal and BPE vocab files with the Normed Bahdanau attention model

Table 8 presents the results obtained by taking train and dev data from the Seen categories (20K data) and testing it for the unseen categories. Since, the paper has defined unseen categories to be those categories for which the corresponding entities are not mapped in the dataset, the delexicalisation dictionary for the 20k dataset is used as it does not contain the mapping of the entities to categories for the unseen test data. It is observed that the BLEU score of the test data after relexicalisation is better at 8.34 in comparison to 6.13 BLEU score of the baseline NMT Model for unseen data.

Dropout	**Encoder Type**	**Attention model**	**Number of Layers**	**BLEU Scores**	**Data**	**Vocab file**
0.5	Unidirectional	Normed Bahdanau	2	test bleu after relex =8.34 test bleu of delex data = 4.5	20k for train and dev, unseen test for testing	normal

Table 8. Results - BLEU scores for unseen category of data.

Experiments were conducted with 4 layers as parameters in the uni-directional RNN to study if there has been any improvement in the BLEU scores, by increasing the number of layers in the recurrent neural network. For the 20k dataset, the parameters for training with 2 and 4 layers have been considered the same, which can be observed in the 1st entry in Table 5 and the entry in Table 9 respectively. The 1st entry in Table 5 is reproduced in Table 9 for the purpose of comparison. The parameters are a drop-out rate of 0.3, by using unidirectional recurrent neural network with standard attention model and the vocabulary file is processed in the traditional manner as described in Section 2.2, where each unique word in the triples and lexicalisations are in each line for their corresponding vocabulary files. The experimental results as shown in Table 9, demonstrate that the BLEU score for the test file after relexicalisation improves to 36.74 with 4 layers of uni-RNN from 34.68 with 2 layer of uni-RNN.

Dropout	**Encoder Type**	**Attention model**	**Number of Layers**	**BLEU Scores [delex]**	**BLEU Scores [relex]**	**Data**	**Vocab file**
0.3	Unidirectional	Standard attention model	2	dev bleu = 19.7 test bleu = 18.1	dev bleu = 38.83 test bleu = 34.68	20k	normal
0.3	Unidirectional	Standard attention model	4	dev bleu = 19.9, test bleu = 18.2	dev bleu = 38.04 test bleu = 36.74	20k	normal

Table 9. Results - BLEU scores with 2 layer and 4 layer of unidirectional RNN

As can be seen from the Table 10 below our scores on Seen Categories (20K data) are on par with the lower end of the results obtained by the other challenge teams. However, on All Categories (22K data) are system has comparable results to the other challenge teams. The score for unseen category is not reported in the table as the parameters for training the data on seen categories in Tables 5,6 and 7 and the unseen categories in Table 8 are different. However, the BLEU score of the 22k dataset can also be considered. The 22k dataset was created by aggregating the provided unseen test data with the provided train and development folders. Our delexicalisation procedure for both seen and unseen categories in the 22k dataset involves retrieval of categories from DBpedia by WebNLG instead of a pre-defined delexicalisation dictionary that was provided in the WebNLG challenge.

Serial No.	System	BLEU Score (All Categories)	BLEU Score (Seen Categories)	BLEU Score (Unseen Categories)
1	ADAPT	31.06	60.59	10.53
2	Melbourne	45.13	54.52	33.27
3	Tilb-SMT	44.28	54.29	29.88
4	Baseline	33.24	52.39	6.13
5	PKUWriter	39.88	51.23	25.36
6	Tilb-NMT	34.60	43.28	25.12
7	UPF-FORGe	38.65	40.88	35.70
8	**Our Model**	**41.5**	**42.67**	**-**
9	UIT-VNU	7.07	19.87	0.11

Table 10. Results - Comparison of BLEU scores of our system with the WebNLG 2017 challenge systems.

6. Discussion

Several experiments have been conducted in Phase 2 of the project to demonstrate the influence of the various improvements in the process of delexicalisation, introduction of the process of relexicalisation, dropout rates, attention models and vocabulary which includes normal vocabulary and Byte Pair Encoding (BPE) vocabulary. The following section include discussions on improvements in BLEU scores obtained as a result of the introduction of the parameters as mentioned.

The BLEU scores have improved significantly. After delexicalisation, the average BLEU score of the test data for experiments is around 20, while after relexicalisation it grows to around 40. In comparison, the BLEU score of the test data in Phase 1 was 12.2 after delexicalisation. When training with delexicalisation, the model finds not only the correlation between nearby words, but also the specific structures for expressing information from a category. When replacing the category with entities in relexicalisation, only objects of a specific type would be considered as choices, so the hit rate would be much higher than selecting a word from thousands of vocabularies. It works especially well when the target is a proper noun that is not used frequently and has low correlations with other words, which is exactly the case of data for WebNLG challenge.

From the results in the tables 5, 6 and 7, we can see that a higher dropout rate gives either a similar or higher accuracy compared to that of a lower drop out rate. In the case that testing uses attention models (either scaled luong or normed bahdanau), both the BLEU scores after delexicalisation and relexicalisation have little difference with either drop out rates(0.3 or 0.5). It is same with BLEU scores after delexicalisation with the standard attention mechanism. However, the BLEU scores after relexicalisation for a model with the standard attention mechanism always increases when the dropout rate increases from 0.3 to 0.5. A higher dropout rate helps us avoid overfitting and learn from different training samples evenly, which would improve the accuracy of the model. It is useful in our NLG task as a chain of triples can have multiple lexicalisations, due to which the same triples are repeated in the source side of the training task.

By comparing the results from tables 5, 6, and 7, we can see that the accuracy increases drastically when we use the attention mechanisms of scaled luong and normed bahdanau. For a model with standard attention mechanism, the BLEU after delexicalisation is always below 20 and the BLEU after relexicalisation is always below 40, while those with attention models exceed the limestone of 20 and 40 easily. Scaled-luong generates larger differences between BLEUs of test data and dev data. The difference between the dev and test BLEU scores for the same training process is always smaller than that for the normed-bahdanau attention mechanism. However, such difference can be as large as 5 for the scaled-luong attention mechanism. However, the overall performance of scaled-luong is better than that of normed-bahdanau. The BLEU score of processes with normed-bahdanau have an average BLEU less than that of scaled-luong by 3. The test BLEU score after relexicalisation with scaled-luong can approach higher than 40, while that with normed-bahdanau cannot. The dev BLEU score of scaled-luong can even be as high as 45. The factor that brings such advantage is that Luong takes the values of the current hidden layer into context vector for prediction. In this case, the process is more concise and the prediction will be less affected by the old patterns, and thus can also avoid overfitting. Another advantage of Luong is that it has more alignment functions than Bahdanau does, so it can always choose the function that is most suitable for the current case.

As is evident from the results in the previous section, the BLEU scores for the BPE vocabulary files are higher than the scores obtained from testing the data with the normal vocabulary files. These results are consistent with section 2.2 which states that the BPE improves the vocabulary and hence the BLEU scores. Our provided dataset was smaller, than the dataset of WebNLG, which we tried to overcome by aggregating the unseen test data with the provided data in the train and dev folders. It has been realized that large amount of training data is necessary to apply neural architectures to NLP tasks.

7. Conclusion

Based on the weaknesses of the baseline system identified in the first phase of the project, we have modified our logic for delexicalisation and implemented relexicalisation of the delexicalized data and have obtained satisfactory results. Delexicalisation and relexicalisation provides the greatest increase to the performance of the model, producing a significant increase in the BLEU scores. This performance is ultimately allowed by the delexicalisation step, which allows generalisation of the data. Our improvements to the delexicalisation model has shown improvements to the performance. By mandating that all entities should have a delexicalisation has allowed further generalisation. The implementation of relexicalisation, which was not present in phase 1, was, of course, a necessary component of the increase obtained by delexicalisation.

Furthermore, we have seen that the use of an attention model can lead to a further increase in performance. Of the three attention models considered, it appears that the scaled Luong attention model, on average, provides the greatest increase in performance. The attention model as described above in Section 4 allows increased BLEU score by connecting parts of the sentence and identifying keywords, which leads to better translation, especially on long sentences.

A few other features which increased performance by small amounts was use of BPE for vocabulary generation and use of a dropout rate of 0.5 rather than 0.3. Possible avenues of research that can be explored have been presented in the Future Work Section.

8. Future Work

One improvement would be to further automate the selection of entity category information for the delexicalisation dictionary. The judgement calls made during this process may have lead to unuseful categorizations that were too specific to generalize (e.g. categories which only had one entity). The proposed model would be able to read in all the entities and their type information, and decide on category placements in such a way to minimize over-specification (placing entities into categories where they are the only one) and overgeneralization (placing too many entities in some common base class). This tradeoff would be a parameter of the model.

Another improvement would be introducing a preprocessing step which could read in triple sets and produce a string of entities and predicates which more closely models natural language. An example of such a system would be simple aggregation. In this model, each subject is produced only once, followed by all the predicate-object pairs which had that subject. This approach addresses the aggregation aspect of natural language. Another approach would be to train a model to automatically organize entities and predicates into a ordering which more closely aligns with natural language. The model could be trained on the triple-lex pairs already provided in this challenge. The goal of such a system is to increase alignment between the source and target, which should increase the performance of the sequence-to-sequence NMT models used.

The delexicalisation procedure can be improved from the replacement of entities by categories by direct matching of the tokenized entities to ensure matching by some similarity metrics. Current work generalizes the semantic information by using the delexicalisation and relexicalisation procedure while generating lexicalisations from RDF triples. This can be improved by aggregating similar categories in order to capture the syntactic information of the mapping of triples to corresponding lexicalisations. Certain rules can be devised to map the triples in the source side to the lexicalisations on the target side before training the dataset which will also help in capturing the alignment of a number of triples to several sections of a lexicalisations. Many categories are missing in the generated lexicalisation file by the NMT Model. Rules can be devised to add the missing categories to the lexicalisation during post-processing. Further experiments can be done by varying the training parameters to improve the quality of the lexicalisation.

9. References

1. The WebNLG Challenge: Generating Text from RDF Data, Gardent et. al., Proceedings of the 10th International Natural Language Generation Conference, pp. 124-133, Spain, 2017.

2. Handling Rare Items in Data-to-Text Generation, Shimorina and Gardent, Proceedings of the 11th International Natural Language Generation Conference, pp. 360-370, Netherlands, 2018.

3. Bahdanau, D., Cho, K. and Bengio, Y. (2015). Neural Machine Translation by Jointly Learning to Align and Translate. CoRR.

4. Salimans, T., and Kingma, D.P. (2016). Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks.

5. https://github.com/rsennrich/subword-nmt

6. https://www.nltk.org/