TRIPLES2TEXT

ARI2201 – Individual Assigned Practical Task

Kelsey Bonnici

Supervisor: Dr Claudia Borg

Table of Contents

Introduction

This project deals with the challenge of converting structured data into coherent and natural language verbalisations. Using an LSTM artificial neural network, the mapping from RDF-Triples to text was investigated in both English and Maltese. In addition to this, the use of GloVe embeddings in the English model was also investigated.

Literature

This project is based on the WebNLG Challenge which consists of mapping data to text. This involves a number of natural language generation subtasks such as sentence segmentation, lexicalisation, aggregation, and surface realisation [1].

Sentence segmentation involves dividing the input data into separate sentences. Since the input data consists of multiple triples, sentence segmentation determines how these triples are grouped together to form coherent sentence in the text [2].

Lexicalisation involves converting the properties of the data into their corresponding natural language expressions. This ensures that the properties are represented in a human-readable form in the output text. A property could be lexicalised as a verb, a relational noun, a preposition, or an adjective [3].

Aggregation aims to avoid repetition in the text. It ensures that information is not unnecessarily repeated. This helps in creating concise and coherent text by combining related information. The need for this arises main when an entry contains more than one triple [3].

Surface realisation involves constructing syntactically correct and natural-sounding sentences following a given grammar [4]. The goal is to generate fluent and well-formed sentences that accurately represent the given data.

The creation of the WebNLG dataset and this challenge aimed to encourage the development in two key areas: RDF verbalisers and micro planners which are capable of handling a range of linguistic constructions [1].

The RDF language used in DBpedia, and other Linked Data datasets is widely used. Many large-scale datasets such as MusicBrainz, FOAF and LinkedGeoData are formatted in the same manner. Generating high-quality text from RDF data would enhance accessibility for non-experts, enrich existing text with knowledge from such databases, and facilitate the description, comparison, and correlation of entities present in these knowledge bases [1].

The 2023 edition of the WebNLG challenge also includes four low resource languages which are lacking in research in text generation. These include Maltese, Irish, Breton and Welsh [5]. Research in low resource languages helps promote technological advancements in these languages further improving their overall accessibility. It fosters the development of more inclusive and diverse NLG technologies.

In the past, data-to-text generation relied on rule-based systems with modular pipeline architectures [6]. However, due to recent developments, there has been a shift towards end-to-end architectures that utilise neural networks to directly convert input data into natural language output. This shift is influenced by the success of end-to-end approaches in machine translation and the availability of large datasets [7].

There are however multiple approaches to such this problem. Below are some descriptions of systems which have been presented for the 2020 WebNLG Challenge [8].

Mono-lingual, mono-task, template-based approaches in NLG involve generating text in a single language for a specific task using predefined templates. These approaches rely on filling placeholders in the templates with relevant data to create the final output. In [9] such an approach is implemented. The input RDF triples are divided and ordered into subsets of sentences. Each subset is then transformed into a sentence using Python procedures that incorporate 200 manually defined sentence templates. Aggregation is managed by combining templates, and referring expressions are generated using names for first occurrences and pronouns for subsequent occurrences within a template. The resulting sequence of sentence templates is mapped to actual sentences using the REAL surface realiser [8].

Mono-lingual, mono-task, neural approaches generate text in a single language for a task using neural network models. These are designed for a particular task, trained on large datasets, and leverage the power of neural networks to produce high quality text. This approach is adopted in [10] where the BART [11] model is employed as their base model. It is trained first on a Wikipedia corpus of 57 million sentences and then on a noisy RDF/English text corpus created by applying Open Information Extraction to the collected sentences. For fine-tuning, they experiment with curriculum learning, varying the number of triples which are input. The results show that pre-training and data augmentation helps improve performance. However, a drop in performance is observed when employing curriculum learning [8].

Mono-task, bilingual approaches focus on a single task while operating in two different languages. The system in [12] uses this approach where the mBART [13] model is pre-trained for multilingual denoising using the CC25 corpus extracted from Common Crawl which contains data in 25 languages. The noise function employed by mBART replaces text spans of varying lengths with a mask token, affecting 35% of the words in each instance, and also shuffles the order of sentences. To generate in both English and Russian, two separate mBART models were trained on the RDF-to-English and RDF-to-Russian datasets [8].

Bi-directional monolingual approaches generate text in both forward and backward directions within a single language. These models consider both preceding and succeeding context, improving the quality of the generated text. In [14], the system uses a weakly supervised approach for learning generation and semantic parsing models by bootstrapping from text and RDF data. The T5 pre-trained sequence-to-sequence model is employed to bootstrap the generation model. For semantic parsing, an entity extraction model [15] is utilised to identify entities in the input text and a multi-label classifier to predict relations between entity pairs. Both the input text and the input graph are aligned with their back-translated versions to create aligned training data. The models are iteratively optimised by alternating between the two. The data that is used for bootstrapping is shuffled to ensure non-parallel alignment between the text and RDF in each dataset instance [8].

Bi-directional, bilingual approaches generate text in multiple languages while considering bidirectional context within each language. In the system shown in [16] the T5 pre-trained model is used to investigate the effectiveness of multilingual multi-task learning in both pre-training and fine-tuning stages. For pre-training, the best result is achieved by training T5 on English and Russian Wikipedia data and further training it on the WMT English/Russian parallel corpus. In the fine-tuning phase, they compare different approaches, including monolingual models, bilingual models with multi-tasking on both languages, and bilingual models fine-tuned on a corpus derived from the WebNLG+ dataset by aligning English and Russian sentences and entities. It was observed that the latter model demonstrates significant improvement particularly in handling unseen relations [8].

For this project, one of the approaches found in [7] was followed. Two types of models were tested.

The first being a vanilla sequence-to-sequence LSTM (Long short-term memory) with attention based on the WebNLG baseline system [3]. The parameters for this model were kept at default which are: two hidden layers with 500 LSTM units per hidden layer and word embeddings of 500 dimensions. Droput is applied at 0.3 and the training is done with stochastic gradient descent which starts at a rate of 1.0 with learning rate decay being enabled [7].

The second model is based on the transformer architecture similar to the end-to-end architecture described in [17]. The model follows an encoder-decoder architecture with 6 layers and 512 hidden units. The word embeddings have a dimensionality of 512, and the feed-forward sublayers have a dimensionality of 2048. Each multi-head attention sublayers consists of 8 attention heads. Dropout with a value of 0.1 is applied, and the model is trained using the Adam optimiser [18].

Along with the results from these two models, the authors also use Knowledge Graph Embeddings (KGEs). These embeddings are trained on around 4.2 million entities and 661 relations which are represented in the entire DBpedia repository. To compare the results of the KGEs, the authors also make use of 300-dimensional GloVe embeddings as pre-trained textual embeddings.

For each of the models, there are results compared with and without a delexicalization step. During the data preparation step, the RDF-triple is modified in a similar way as is found in [3]. The subject of the RDF-triple is replaced by the DBpedia category, and the object is replaced by the predicate using a pre-defined delexicalisation dictionary[1].

[1] https://gitlab.com/webnlg/webnlg-baseline/

Data

The data used for this project is a subset of the WebNLG Corpus[2]. Each entry contains a triple/set of triples and a list of corresponding text verbalisations. These verbalisations come in both English and Maltese.

Triple: [Aarhus_Airport | cityServed | Aarhus, Denmark]
Lex 1: The Aarhus is the airport of Aarhus, Denmark. Lex 2: Aarhus Airport serves the city of Aarhus, Denmark. Lex 3: L-Aarhus huwa l-ajruport ta' Aarhus, id-Danimarka. Lex 4: L-ajruport ta' Aarhus isservi l-belt ta' Aarhus, id-Danimarka.

To include a variety of lexicalisation patterns, the data was extracted from 15 DBpedia Categories: Astronaut, University, Monument, Building, ComicsCharacter, Food, Airport, SportsTeam, WrittenWork, Athlete, Artist, City, MeanOfTransportation, CelestialBody and Politician. This resulted in a set of 373 RDF properties [3].

For this project, only a subset of the data provided was used and the data was processed using the provided utilities that came with the data found in *benchmark_reader.py*. This being only entries where there is only a single triple (such as the one above). Each entry for the dataset is created by associating each triple with each sentence and then splitting them by language. At the time of writing, only a training and a development set were made available for use. Therefore, the testing set was created from the training set taking care to not have triples in the test data which also show in the training data. Another issue with the data was also that the number of validation sentences available for Maltese was half of what was available in English. To combat this discrepancy, some data was taken from the training set in the same way as was done for the testing set. The following is the final number of triple-sentence pairs for both English and Maltese.

	English	**Maltese**
Training	5681	4536
Validation	961	853
Testing	769	769

As part of the data pre-processing, multiple ways of formatting the data was tested. The best results were given when the predicate in the triple was split by *camelCase.* As seen in the example above 'cityServed' is changed into 'city Served'. Other methods such as decapitalising all letters and removing special characters were also tested but these proved to give worse results. These results will be further discussed in the evaluation section.

GloVe Embeddings

GloVe is an unsupervised learning algorithm designed to obtain vector representations for words. By utilising aggregated global word-word co-occurrence statistics from a corpus

[2] https://github.com/WebNLG/2023-Challenge

during training, GloVe generations representations which showcase interesting linear substructures of the word vector space [19].

The GloVe model is trained on the non-zero entries of a global word-word co-occurrence matrix, which records the frequency of word co-occurrences in a corpus. Constructing this matrix requires that the corpus is processed in its entirety at least once to collect its statistics. For large corpora, this can be computationally demanding but it is a one-time expense which provides good results. After that, training iterations are much faster since the number of non-zero matrix entries is usually much smaller than the total length of the corpus [19].

For this project, for the English segment, 300-dimensional GloVe embeddings[3] were engaged. These were trained on a combination of a 2014 Wikipedia dump, which has 1.6 billion tokens, and Gigaword5, which has 4.3 billion tokens. Together they total to almost 6 billion tokens. The final result has a vocabulary of 400,000 words [20].

Unfortunately, for the Maltese language, there are not any GloVe embeddings available for use. The Maltese language faces a challenge in projects like these due to it being a low-resource language. The lack of data poses a substantial obstacle in the development of language technologies and natural language processing in Maltese.

Implementation

The implementation was done using the OpenNMT library which is an open-source ecosystem used for neural machine translation and neural sequence learning [21]. Using this library an LSTM (Long Short-Term Memory) model was implemented.

The LSTM model is a type of Recurrent Neural Network (RNN) which is used to identify patterns within sequential data such as those in sensor data, stock prices, or natural language. RNNs have the ability to do this because apart from storing the actual value, they also store its position within the sequence when making a prediction [22].

One limitation that RNNs suffer from is short-term memory where the ability to retain previous information diminishes rapidly as the sequence length increases. To overcome this issue, LSTM models were introduced to enhance the retention of past information for longer periods. It addresses the issue by preserving the relevant information in a long-term memory called the Cell State. Additionally, the model also maintains a hidden state where short-term information from previous computation steps is stored. The hidden state serves as the model's short-term memory [22].

During each step, the LSTM model considers the current input $x(t)$, the previous state of the short-term memory $c(t-1)$, and the previous state of the hidden state $h(t-1)$ [22].

[3] https://nlp.stanford.edu/projects/glove/

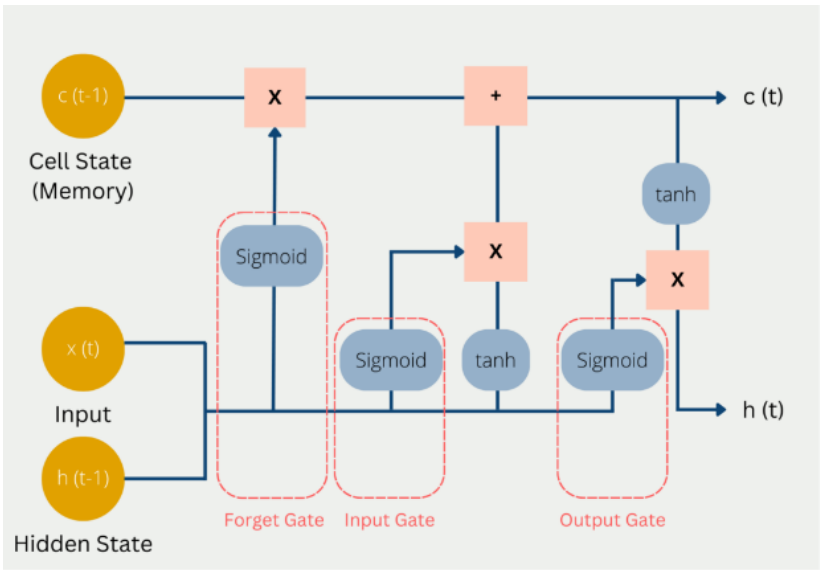

Figure 1: LSTM Architecture

The information flow is controlled by three gates: the Forget Gate, the Input Gate, and the Output Gate. These gates determine which information is retained and passed to the new Cell State and Hidden State [22].

The Forget Gate decides which current and previous information is kept and which is forgotten. It takes the previous hidden state $h(t-1)$ and the current input $x(t)$ and passes them through a sigmoid function, producing values between 0 and 1. The value 0 indicates that certain information can be forgotten as there may be more important information to remember. A value of 1 indicates that the information should be preserved. The resulting values are multiplied by the current Cell state, effectively discarding irrelevant knowledge [22].

The Input Gate assess the relevance of the current input for solving the task at hand. It multiplies the current input with the hidden state $h(t-1)$ and the weight matrix from the previous step. The information deemed important by the Input Gate is added to the Cell State, contributing to the formation of the new Cell state $c(t)$. This updated Cell State becomes the current state of the long-term memory and will be used in the next step [22].

The Output Gate calculates the output of the model in the Hidden State. The sigmoid function determines which information can pass through the Output Gate, and the Cell State is multiplied by the output after being activated by the tanh function [22].

The model was implemented with the default parameters as found in the library. These parameters are two hidden layers with 500 LSTM units per hidden layer and word embeddings of 500 dimensions. Dropout is also applied at 0.3 and the training is done using stochastic gradient descent which starts at a rate of 1.0 and has learning rate decay enabled. These parameters are also the same as one of the systems that was described in [7].

Three different models in total were trained: English, English with GloVe embeddings and Maltese. The vocabulary for each model was built using the tokenizer that is part of the OpenNMT library.

Evaluation

As a metric to measure results, the BLEU (Bilingual Evaluation Understudy) was used. This algorithm is used to assess the accuracy of machine-translated text in comparison to human translations. It measures the quality of the machine's output by comparing it to the provided translations [23]. A script to execute this algorithm on the results was provided along with the dataset.

While looking at the model's output, the biggest problem lies in unknown words. These are words that do not appear in the training set and therefore are not found in the model's vocabulary. Names and numbers seem to make up most of the unknown words.

Below are some examples of the actual triple, their translation, how the model interpreted the triple, and the machine's output along with some comments on the translation.

English

Actual Triple	Alfred_Garth_Jones death Place Sidcup
Actual Translation	Alfred Garth Jones died in Sidcup.
Interpreted Triple	['Alfred_Garth_Jones', 'death', 'Place', '<unk>']
Machine Translation	Alfred Garth Jones died in New York.

This is a case of an unknown word. In this case it is 'Sidcup' which is a name of a place. However, the model still managed to infer that a place fits in the sentence therefore generating a sentence which makes sense and is grammatically correct although not factually correct.

Actual Triple	Canada official Language French_language
Actual Translation	French is Canada's official language.
Interpreted Triple	['Canada', 'official', 'Language', 'French_language']
Machine Translation	The official language of Argentina is Spanish.

In this example, there are no unknown words. The output is still about the official language of a country, but it is not about Canada. The sentence is still both grammatically and factually correct.

Maltese

Actual Triple	Texas largest City Houston
Actual Translation	Houston hija l-akbarbelt fit-Tessas.
Interpreted Triple	['Texas', 'largest', 'City', 'Houston']
Machine Translation	Austin hija l-akbarbelt tat-Tessas.

This example is similar to the second example for English. It switched the word 'Houston' with 'Austin'. It is worth noting that Austin is also a city in Texas. In the actual translation of this example, it can be observed that there is a missing space between the words 'l-akbar' and 'belt'.

Actual Triple	Ska_punk stylistic Origin Ska
Actual Translation	Ska punk li ġej minn mużika sk.
Interpreted Triple	['Ska_punk', 'stylistic', 'Origin', 'Ska']
Machine Translation	L-oriġini stilistika hija l-oriġini stilistika stilistika tal-Ajru tal-Ajru rock.

This example shows a case where the model is repeating words.

Overall, most sentence from all three models are legible and few sentences contain a lot of repeating words. In the case of Maltese, one of the commonly observed problems is in the case of gendered nouns. This is not an issue in English as the English language does not have gendered nouns.

As was discussed in the data section, different ways of formatting the data was tested to see what kind of results would be produced. Formatting data in different ways affects the vocabulary that the system will use.

1. Data is kept as is.

Triple	['Aarhus_Airport', 'cityServed', '"Aarhus, Denmark"']
Sentence in English	The Aarhus is the airport of Aarhus, Denmark.
Sentence in Maltese	L-Aarhus huwa l-ajruport ta' Aarhus, id-Danimarka.

2. Regular expression is used to remove special characters in triple.

Triple	['Aarhus Airport', 'cityServed', ' Aarhus Denmark ']
Sentence in English	The Aarhus is the airport of Aarhus, Denmark.
Sentence in Maltese	L-Aarhus huwa l-ajruport ta' Aarhus, id-Danimarka.

3. Everything is set to lowercase.

Triple	['aarhus_airport', 'cityserved', '"aarhus, denmark"']
Sentence in English	the aarhus is the airport of aarhus, denmark.
Sentence in Maltese	l-aarhus huwa l-ajruport ta' aarhus, id-danimarka.

4. Predicate is split into separate words.

Triple	['Aarhus_Airport', 'city Served', '"Aarhus, Denmark"']
Sentence in English	The Aarhus is the airport of Aarhus, Denmark.
Sentence in Maltese	L-Aarhus huwa l-ajruport ta' Aarhus, id-Danimarka.

5. RE + Lowercase

Triple	['aarhus airport', 'cityserved', ' aarhus denmark ']
Sentence in English	the aarhus is the airport of aarhus, denmark.
Sentence in Maltese	l-aarhus huwa l-ajruport ta' aarhus, id-danimarka.

6. RE + Splitting predicate

Triple	['Aarhus Airport', 'city Served', ' Aarhus Denmark ']
Sentence in English	The Aarhus is the airport of Aarhus, Denmark.
Sentence in Maltese	L-Aarhus huwa l-ajruport ta’ Aarhus, id-Danimarka.

7. Lowercase + Splitting

Triple	['aarhus_airport', 'city served', '"aarhus, denmark"']
Sentence in English	the aarhus is the airport of aarhus, denmark.
Sentence in Maltese	l-aarhus huwa l-ajruport ta’ aarhus, id-danimarka.

8. RE + Lowercase + Splitting

Triple	['aarhus airport', 'city served', ' aarhus denmark ']
Sentence in English	the aarhus is the airport of aarhus, denmark.
Sentence in Maltese	the aarhus is the airport of aarhus, denmark.

The following is a table of all the BLEU scores obtained for English, English with GloVe embeddings and Maltese. The result highlighted in bold is the best one from each.

	1	2	3	4	5	6	7	8
English	16.4104	13.3489	14.6464	**20.1420**	14.9612	15.3697	15.3500	15.6839
English w/GloVe	**18.0305**	13.7654	16.2314	17.4939	12.2937	13.3875	17.7520	14.3805
Maltese	7.34072	9.46474	7.63310	4.71897	7.90859	8.98513	8.98513	**9.68978**

From these results it was decided that the only data formatting that is done is splitting the predicate into different words. Although this method did not give the highest score for the English model with the GloVe embeddings, there is a small enough difference in the score.

In the case of Maltese, this data formatting gave a very low score. It is not clear why there is such a huge discrepancy where the other formatting methods lay in the range of 7 to 9. There is also a huge discrepancy between the results for both English models and the Maltese one which will be discussed later.

It was expected that the English model with the GloVe embeddings would give higher scores than that of the model without. It is not clear why this was not the case overall.

Regarding the Maltese results, there are a couple of reasons as why the score could be so low.

1. The tokeniser from the OpenNMT library is not adapted for Maltese. The different grammar rules and sentence structure from English would result in the vocabulary for Maltese not being built correctly. Such an issue could be avoided by using a tokeniser that is adapted for Maltese.

2. The triples being fed into the model are in English and were not translated into Maltese beforehand. This could cause an issue.
3. The translated sentences in Maltese sometimes contain mistakes as seen in the first example. Mistakes in the data such as these will result in the model not learning correctly and making similar mistakes.

Conclusion

This project investigated the conversion of structured data into natural language verbalisation using LSTM models. This was investigated in both English and Maltese, and also investigated the use of GloVe embeddings in the English model. The results showed coherent and appropriate output for the most part. Overall, such a project contributes to the advancement of the task of converting structured data into sentences. In addition to this, including low-resource languages also helps further research into them.

References

[1] “What is WebNLG Challenge?,” WebNLG Challenges, [Online]. Available: https://synalp.gitlabpages.inria.fr/webnlg-challenge/.

[2] V. Yordanov, “Introduction to Natural Language Processing for Text,” Medium, 17 November 2018. [Online]. Available: https://towardsdatascience.com/introduction-to-natural-language-processing-for-text-df845750fb63.

[3] C. Gardent, S. Anastasia, N. Shashi and L. Perez-Beltrachini, “The WebNLG Challenge: Generating Text from RDF Data,” in *Proceedings of the 10th International Conference on Natural Language Generation*, Santiago de Compostela, Spain, 2017.

[4] A. Shimorina and C. Gardent, “Surface Realisation Using Full Delexicalisation,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China, 2019.

[5] “WebNLG Challenge 2023,” WebNLG Challenges, [Online]. Available: https://synalp.gitlabpages.inria.fr/webnlg-challenge/challenge_2023/.

[6] A. Gatt and E. Krahmer, “Survery of the State of the Art in Natural Language Generation: Core tasks, applications and evaluation,” *Journal of AI Research (JAIR),* vol. 61, pp. 75-170, 2018.

[7] N. Pasricha, M. Arcan and P. Buitelaar, “Utilising Knowledge Graph Embeddings for Data-To-Text Generation,” in *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, Dublin, Ireland (Virtual), 2020.

[8] T. Castro Ferreira, C. Gardent, N. Ilinykh, C. van der Lee, S. Mille, D. Moussallem and A. Shimorina, “The 2020 Bilingual, Bi-Directional WebNLG+ Shared Task: Overview and Evaluation Results (WebNLG+ 2020),” in *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, Dublin, Ireland (Virutal), 2020.

[9] G. Lapalme, “RDFjsRealB: a Symbolic Approach for Generating Text from RDF Triples,” in *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, Dublin, Ireland (Virtual), 2020.

[10] S. Montella, B. Fabre, T. Urvoy, J. Heinecke and L. Rojas-Barahona, “Denoising Pre-Training and Data Augmentation Strategies for Enhanced RDF Verbalization with Transformers,” in *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, Dublin, Ireland (Virtual), 2020.

[11] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov and L. Zettlemoyer, “BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online, 2020.

[12] Z. Kasner and O. Dušek, “Train Hard, Finetune Easy: Multilingual Denoising for RDF-to-Text Generation,” in *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, Dublin, Ireland (Virtual), 2020.

[13] Y. Liu, J. Gu, N. Goyal, X. Li, S. Edunov, M. Ghazvininejad, M. Lewis and L. Zettlemoyer, “Multilingual Denoising Pre-training for Neural Machine Translation,” 2020.

[14] Q. Guo, Z. Jin, X. Qiu, W. Zhang, D. Wipf and Z. Zhang, “CycleGT: Unsupervised Graph-to-Text and Text-to-Graph Generation via Cycle Training,” in *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, Dublin, Ireland (Virtual), 2020.

[15] P. Qi, Y. Zhang, Y. Zhang, J. Bolton and C. D. Manning, “Stanza: A Python Natural Language Processing Toolkit for Many Human Languages,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2020.

[16] O. Agarwal, M. Kale, H. Ge, S. Shakeri and R. Al-Tfou, “Machine Translation Aided Bilingual Data-to-Text Generation and Semantic Parsing,” in *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, Dublin, Ireland (Virtual), 2020.

[17] T. Castro Ferreira, C. van der Lee, E. van Miltenburg and E. Krahmer, “Neural data-to-text generation: A comparison between pipeline and end-to-end architectures,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China, 2019.

[18] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” in *3rd International Conference for Learning Representations*, San Diego, 2015.

[19] J. Pennington, R. Socher and C. Manning, “GloVe: Global Vectors for Word Representation,” Stanford Edu, August 2014. [Online]. Available: https://nlp.stanford.edu/projects/glove/.

[20] J. Pennington, R. Socher and C. Manning, “GloVe: Global Vectors for Word Representation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 2014.

[21] G. Klein, Y. Kim, Y. Deng, J. Senellart and A. Rush, "OpenNMT: Open-Source Toolkit for Neural Machine Translation," in *Proceedings of ACL 2017, System Demonstrations*, Vancouver, Canada, 2017.

[22] "Long Short-Term Memory Networks (LSTM) - simply explained!," Data Base Camp, 4 June 2022. [Online]. Available: https://databasecamp.de/en/ml/lstms.

[23] K. Papineni, S. Roukos, T. Ward and W.-j. Zhu, "Bleu: a method for Automatic Evaluation of Machine Translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, Pennsylvania, USA, 2002.

Plagiarism Declaration Form

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Declaration

Plagiarism is defined as “the unacknowledged use, as one's own, of work of another person, whether or not such work has been published, and as may be further elaborated in Faculty or University guidelines" (University Assessment Regulations, 2009, Regulation 39 (b)(i), University of Malta).

I, the undersigned, declare that the assignment submitted is my work, except where acknowledged and referenced.

I understand that the penalties for committing a breach of the regulations include loss of marks; cancellation of examination results; enforced suspension of studies; or expulsion from the degree programme.

Work submitted without this signed declaration will not be corrected and will be given zero marks.

Kelsey Bonnici	[illegible]
Student Name	Signature
ARI2201	Triples2Text
Course Code	Title of work submitted
11/06/2023	
Date	