Sequence-to-Sequence Modelling for RDF triples to Natural Text

David Bergés[1], **Roser Cantenys**[2], **Roger Creus**[3], **Oriol Domingo**[4]
Universitat Politècnica de Catalunya, Barcelona
{david.berges[1], roger.creus[3], oriol.domingo.roig[4]}@est.fib.upc.edu
{roser.cantenys[2]}@upc.edu

Abstract

This work presents a novel case of Machine Translation (MT) approach to a different domain: data-to-text. Specifically, we are establishing some traits on how, which and when MT techniques are worth applying to RDF-to-Text task. Not only did we apply most prominent MT architectures, such as the Transformer, but we also analyzed state-of-the-art techniques such as Byte Pair Encoding or Back Translation. In addition, we empirically show how to tailor these techniques to improve models relying on learned embeddings rather than using pretrained ones.

1 Introduction

A Knowledge Base (KB) is a large source of information represented in a structured way. The information structure is based on Resource Description Frameworks (RDF), which consists of three elements: $\langle$*subject, predicate, object*$\rangle$. Thus, it establishes relations (*predicate*) between entities (*subject, object*).

It is known that KB are being widely considered in the industry for applications such as question answering (Q&A) systems (Fader et al., 2014), search engines (Ding et al., 2004), recommender systems (Huang et al., 2002), etc. However, this data representation is not human-friendly, i.e. is not in a language form, hence, it is hard for human to comprehend the information embedded in these triples.

The team address this problem under the context of Natural Language Generation (NLG). This task can be divided into: text-to-text generation or data-to-text generation, according to Gatt and Krahmer (2017). The latter is the one considered since we are mapping structured data to text, however, we also applied Machine Translation (MT) architectures and techniques, which are considered for text-to-text generation.

To further improve the quality of the text generated by our models, we studied the influence of Back-Translation (BT) (Sennrich et al., 2016) for this task, as well as, the influence of learning the embeddings from scratch or providing pretrained embeddings.

Our contributions are:

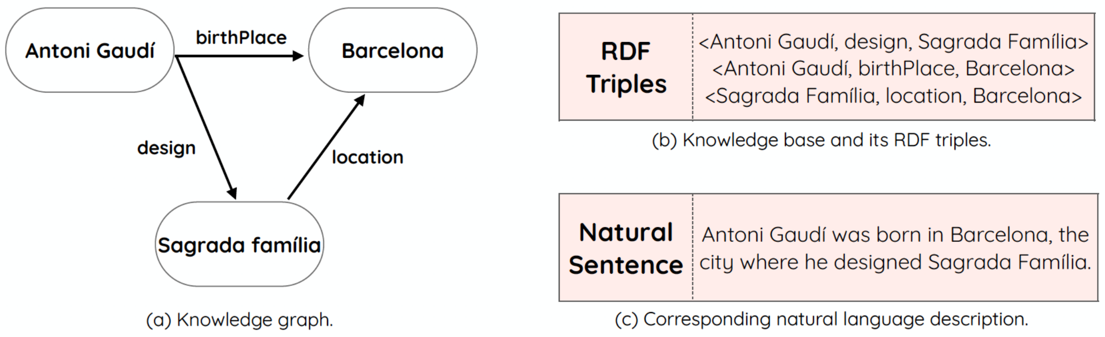
- We propose an encoder-decoder Convolutional architecture for RDF to Text task.
- We propose an encoder-decoder Transformer architecture for RDF to Text task.
- We enhance our models with Byte Pair Encoding (BPE) (Sennrich et al., 2015), embedding analysis and BT.

Therefore, the report starts formulating the task as well as studying the related work in order to propose a system architecture with the specific aforementioned models. Then, we will detail the experiments and results with BPE, embeddings and BT techniques and, finally, conclusions will be presented.

2 Task Formulation

This section presents a formal definition of the problem and provides the mathematical nomenclature used in the whole project.

The input to our system is a KB that can be denoted as a set of RDFs, i.e. $\mathcal{K} := \{r_1, ..., r_n\}$. Each RDF r_i can be defined as $\langle s_i, p_i, o_i \rangle$, these elements stands for subject, predicate and object, respectively. Notice that each element can contain more than one word, there is no prior restriction in that sense. For instance, the subject 'Barack Obama' would be encoded as $s_i = [Barack, Obama] = s_{ij} = [s_{i1}, s_{i2}]$, so i indicates the RDF in $\mathcal{K}$ and



(a) Knowledge graph.

RDF Triples	<Antoni Gaudí, design, Sagrada Família> <Antoni Gaudí, birthPlace, Barcelona> <Sagrada Família, location, Barcelona>

(b) Knowledge base and its RDF triples.

Natural Sentence	Antoni Gaudí was born in Barcelona, the city where he designed Sagrada Família.

(c) Corresponding natural language description.

Figure 1: Example of a knowledge graph (a) with its corresponding knowledge base (b) and its natural language description (c). Design inspired by (Zhu et al., 2019).

another index j is added to denote word position in each subject, predicate or object element.

Finally, we aim to generate a sequence of sentences $\mathcal{S}$, which consists of a sequence of words $[w_1, ..., w_m]$. The resulting sentences in $\mathcal{S}$ should be grammatically correct and should also contain all the information present in the KB $\mathcal{K}$.

In order to guarantee the correctness of $\mathcal{S}$, the following metrics are considered: BLEU (Papineni et al., 2002), METEOR (Banerjee, Lavie, 2005), chrF++ (Popović, 2017) and TER (Snover et al., 2006).

3 Related Work

The studied task was proposed in the WebNLG Challenge 2017[1]. Submissions to this challenge include different approaches, such as: Neural Machine Translation (NMT), Statistical Machine Translation (SMT) and Pipeline systems (Gardent et al., 2017b). The most prominent model regarding automatic evaluation was from the University of Melbourne, Australia, which was based on NMT. The model consists of an encoder-decoder Bidirectional Long Short-Term Memory architecture with attention[2].

Recently, most of the taken approaches to solve this NLG task are based on encoder-decoder architectures as well as Graph Neural Networks (GNN). The main idea behind these methods is to exploit the input structure, which can be seen as a graph: Figure 1. It can be empirically shown how encoder-decoder GNN architectures can achieve similar results to the best submission in WebNLG Challenge 2017 (Marcheggiani, Perez-Beltrachini, 2018) or even improve these benchmarks (Trisedya et al., 2018).

[1]`https://webnlg-challenge.loria.fr`
[2]`https://webnlg-challenge.loria.fr/files/melbourne_report.pdf`

4 System Architecture

This section describes the end-to-end pipeline architecture from preprocessing $\mathcal{K}$ to postprocessing the output of intermediate models.

4.1 Preprocessing

Preprocessing steps are performed sequentially aiming to generalize the content present in $\mathcal{K}$. First of all, each RDF is delexicalise as suggested in the WebNLG Challenge 2017. Table 1 illustrates how one RDF that establishes a relationship between specific entities (`Rome, Italy`) can be generalized to any pair of entities of the same type (`CITY, COUNTRY`). Hence, we also need to delexicalise every target sentence to match the delexicalise input during training phase.

Then, Moses tokenizer (Koehn et al., 2007) is applied to separate punctuation from words, preserving special tokens such as dates, and normalize characters. Finally, it has been studied that BPE is helpful to improve the translation quality. This technique is capable of open-vocabulary translation by encoding rare and unknown words as sequences of subword units (Sennrich et al., 2015), hence, providing more confidence in unseen vocabulary translation. The team applied this technique in order to attain high quality text regardless whether vocabulary has been seen during the train process or not. More details about the advantages and disadvantages of this technique will be discused in Section 6.1.

Type of Text	Lexicalise	Delexicalise
RDF	⟨Rome, capital of, Italy⟩	⟨CITY, capitalOf, COUNTRY⟩
Target Sentence	Rome is the capital of Italy.	CITY is the capitalOf COUNTRY.

Table 1: Lexicalise and delexicalise language.

4.2 Proposed Models

Sequence to sequence modeling has been synonymous with Recurrent Neural Network (RNN) based encoder-decoder architectures (Sutskever et al., 2014; Bahdanau et al., 2015) and later Convolutional Neural Networks (CNN) (Gehring et al., 2017) ever before the appearance of more advanced Transformer-based models (Vaswani et al., 2017).

In the first, the encoder RNN processes an input sequence $x = (x_1, ..., x_m)$ of m preprocessed elements and returns state representations $z = (z_1, ..., z_m)$. The decoder RNN takes z and generates the output sequence $y = (y_1, ..., y_n)$ left to right, one element at a time. To generate output y_{i+1}, the decoder computes a new hidden state h_{i+1} based on the previous state h_i, an embedding g_i of the previous target language word y_i, as well as a conditional input c_i derived from the encoder output z. Based on this generic formulation, various encoder-decoder architectures have been proposed, which differ mainly in the conditional input and type of RNN units (Hochreiter, Schmidhuber, 1997; Chung et al., 2014) being used.

Instead of relying on RNNs, fully convolutional architectures for sequence to sequence modeling have been very popular as well. This ones, on behalf of computing the intermediate encoder states z and decoder states h, make use of fully convolutional layers.

Transformer-based models, however, are model architectures eschewing recurrence and instead rely entirely on attention mechanisms to draw global dependencies between inputs and outputs. The Transformer allows for significantly more parallelization and has shown to be able to reach state-of-the-art results in many Natural Language Processing (NLP) tasks.

4.2.1 Convolutional based

In order to preserve the long-range dependencies between words and its context, we implemented a sequence-to-sequence, encoder-decoder based Convolutional model. This architecture was proposed in (Angela Fan, 2018) to solve a text-to-text NLG task for storytelling. Thus, we find it appropriate to use this model as the starting point or vanilla model in our different NLG task.

This architecture wisely combines the ability of precise context analysis by means of multiple convolutional filters of limited size together with the joint analysis of their results with the attention mechanism.

One-dimensional convolutions are performed in both encoder and decoder for obtaining space-invariant feature vectors that preserve the relation between the words covered by the implicit windowing. This features are assembled with an attention mechanism allowing to extend the context size and preserve even longer range dependencies.

4.2.2 Transformer based

The team also implemented a sequence-to-sequence, encoder-decoder based on the Transformer model proposed in (Vaswani et al., 2017) to empirically demonstrate whether most prominent MT architectures and techniques in a data-to-text domain work or not. Moreover, Transformer can be interpreted as a special case of GNN, mentioned by Chaitanya Joshi[3]. Hence, this approach is also aligned with the latest research, mainly based on GNN, conducted to tackle the problem of RDF-to-text.

This model is implemented with an attention mechanism to allow modeling dependencies regardless their distance in the input or output sequences. This results in a fundamental feature for NLG, since automated generation of text depends on the capability of combining information given by different, and not only consecutive, words.

4.3 Postprocessing

Models output a sequence of predicted words, then, the system removes the Moses tokenization as well as BPE. Moreover, the output will be delexicalise since we input and train the models to do so. Hence, we need to perform a relexicalisation step in order to recover the specific relationships and meanings from the original lexicalise RDF.

[3]https://graphdeeplearning.github.io/post/transformers-are-gnns/

5 Experimental Setup

Here, we describe the dataset used and hyperparameter tuning before presenting experimental results in next Section 6.

5.1 Data

The data used in this work is the `release_v2.1` (Gardent et al., 2017a) taken from the WebNLG corpus[4]. This dataset is partitioned into three subsets: train, dev and test, which respectively have 34338, 4313 and 4222 instances. This data is based on DBPedia, which is a multilingual KB that was built from several kinds of structured information included in Wikipedia (Mendes et al., 2012).

As mentioned before in Section 4.1, data has been delexicalised as suggested in the WebNLG Challenge 2017. Consequently, it can be seen in Figure 2 that the number of unique words in text and in RDF has been greatly reduced aiming to obtain more generalization.

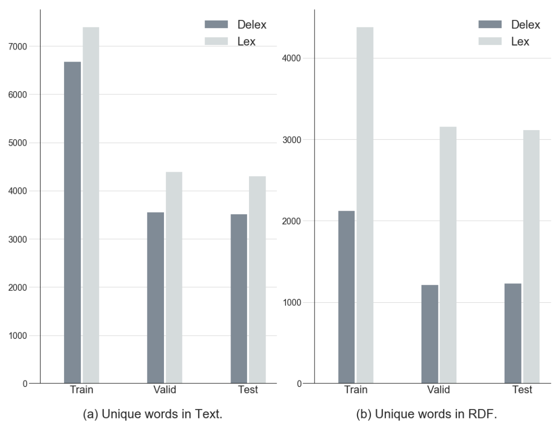


Figure 2: Reduction of the instance number of the three data subsets after delexicalization in text and in RDF.

5.2 Model Parameters and Optimization

This section describes the training regime for the proposed models in Section 4.2.

Both the Transformer and the CNN are primarily based on the same core concept, that is, the usage of attention mechanisms. Thus, they share some of the main architecture parameters, that we used in our initial hyperparameter tuning approach. We performed a grid search on the number of layers, the embedding dimension and the number of attention heads. In the end, the best values were between 3 or 4 layers, 256 and 300 dimensions for the embeddings and 8 attention heads.

[4]https://gitlab.com/shimorina/webnlg-dataset

In addition to the above-mentioned, we further studied the Transformer model, by means of performing another small grid search with the hidden dimension of the Feed Forward Network (FFN), whether to use learned positional embeddings and whether to use cross + self attention throughout the different layers. At last, we obtained the best results using 1024 as the hidden dimension for the FFNs, using fixed sinusoidal positional embeddings and cross attention. In Figure 3 we can see the training and validation losses for the best configurations for both the Transformer and the CNN.

5.2.1 Hardware and Schedule

We trained our models on a single machine equiped with 2 NVIDIA GTX 2080Ti GPUs. In order to speedup training for the initial hyperparameter tuning approach, we used Mixed Precision Training (FP16) (Micikevicius et al., 2017) and both available GPUs. For the final models we trained on a single GPU and normal precision training in pursuance of stability and consistency.

As a reference, the final transformer finished training with just over 1 hour for a total of 20 epochs. The CNN is much more computationally intensive and took well over 3 and a half hours.

5.2.2 Optimizer

We used the Adam optimizer (Kingma, Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. We varied the learning rate over the course of training, increasing it linearly for the first *warmup_updates* = 4000 and decreasing it according to the following inverse square root formula:

$$lr_i = \frac{lr_0 \cdot \sqrt{warmup\_updates}}{\sqrt{i}} \tag{1}$$

5.2.3 Regularization

We studied employing three types of regularization during training:

Dropout: In both models, we apply dropout (Srivastava et al., 2014) to the output of each sublayer before it is fed to the next one. In addition, we reckon it could be interesting to further study the effect of dropout inside the attention weights and the activation functions.

Gradient Clipping: In order to avoid problems with exploding gradients, we renormalized the gradients in the CNN if their norm exceeded 0.1 (Pascanu et al., 2012).

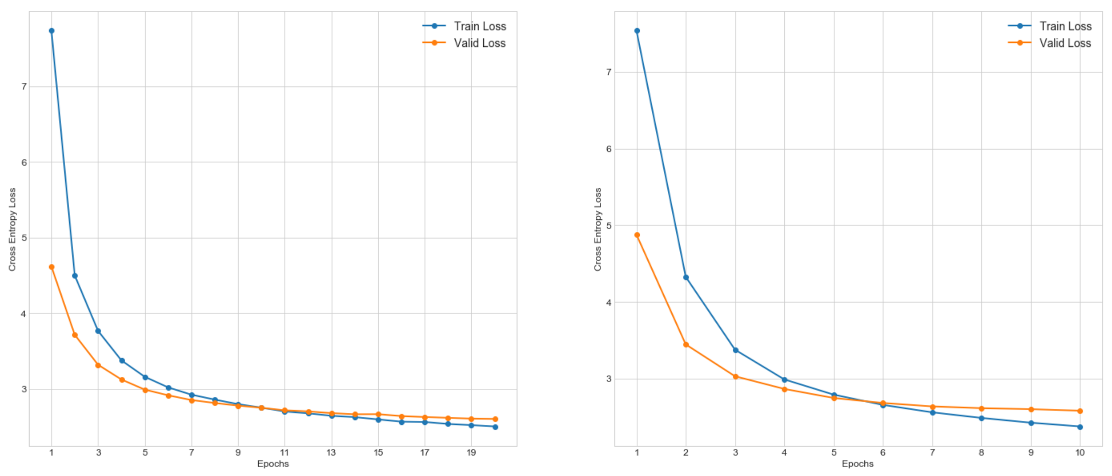


Figure 3: Training and Validation losses for the best performing transformer-based and convolutional-based models respectively. Hence both charts depict approximately the same behaviour, even though the convolutional-based model was only trained for 10 epochs due to computational limitations.

Label Smoothing: During training, we employed label smoothing with $\epsilon_{ls} = 0.1$ (Szegedy et al., 2015). This might have hurt perplexity, as the model learned to be more unsure, but helped to improve some metrics like BLEU score.

6 Experiments and Results

In this section, we present the experiments conducted based on the Transformer model. The first experiment analyses the BPE technique aiming to optimize the number of BPE subwords as well as providing details whether this technique is worth considering as a complementary step to the delexicalisation or not. The second experiment consists of analysing the improvements of pretrained embeddings against learned embeddings during the training process. Last experiment is designed to enlarge training data by means of different BT approaches.

6.1 Byte Pair Encoding

It has been demonstrate that the method used to treat rare items in data-to-text generation strongly impact system performance (Shimorina, Gardent, 2018). They also stated that character-based techniques, such as BPE, obtained very poor results in the WebNLG dataset. Nevertheless, the combination of delexicalisation and BPE were not considered, neither the fact of learning the embeddings from scratch as most MT system do. Thus, we studied the influence of the BPE technique under the use of delexicalisation and learned embeddings of 256-dimensions.

Table 2 shows that not considering BPE, top-row marked as —, could lead to worse performance than using BPE. Nevertheless, the number of BPE subwords needs to be fine-tuned, otherwise, the system's performance could significantly decrease, as in the case of 500 BPE subwords.

One remarkable aspect, is that the test set, from which these metrics were computed, has a vocabulary that mostly appears in the train set. Thus, the performance of systems using BPE technique should be robust to data that consist of unseen vocabulary, while systems that do not consider such a technique would decrease their performance as unknown vocabulary increases, attaining greater differences.

BPE	**BLEU** (↑)	**METEOR** (↑)	**chrF++** (↑)	**TER** (↓)
—	37.15	0.36	0.61	0.55
500	36.41	0.35	0.60	0.53
1000	37.11	0.35	0.61	**0.52**
5000	**38.33**	**0.36**	**0.62**	0.53
7000	37.01	0.35	0.61	0.56

Table 2: Transformer performance regarding different number of BPE subwords.

6.2 Embedding Analysis

In the MT field, the most common approach is to learn embeddings from scratch due to amount of available data. Although this is not our case, in which thousands of instances are provided, we present a comparison between using learned embeddings and pretrained embeddings. As previously mentioned, the dataset consists of instances extracted from Wikipedia, hence, we found that suitable embeddings could be: GloVe (Pennington et al., 2014) and Wikipedia2Vec (Yamada et al., 2020).

The preprocessing and postprocessing pipeline had to be slightly adapted to allow the use of pretrained embeddings. These embeddings rely directly on the lexicalise format as shown in Table 1. Notice that the generalization that delexicalisation step was giving is no longer present using these pretrained embeddings. However, these embeddings have been built using a large amount of data that should lead to good generalization between entities as well.

We mentioned that 256-dimensions for learned embeddings were the best choice after hyperparameter-tuning. Providing this benchmark, we realised that pretrained embeddings were outperforming the learning embedding approach. In Table 3, it is clearly shown how the benchmark can be surpassed regarding any metric even with lower dimensionality, 200-dimension GloVe. Moreover, there is not a notable difference between 200-dimension GloVe, 300-dimension GloVe and 300-dimension Wikipedia2Vec since the small improvements betweem their performance metrics are not significant.

Embedding	**Dimension**	**BLEU** (↑)	**METEOR** (↑)	**chrF++** (↑)	**TER** (↓)
Learned	256	38.33	0.36	0.62	0.53
GloVe	100	36.98	0.35	0.60	0.48
GloVe	200	42.41	**0.39**	**0.66**	0.46
GloVe	300	**42.85**	**0.39**	**0.66**	0.46
Wikipedia2Vec	100	37.45	0.35	0.61	0.48
Wikipedia2Vec	300	41.69	**0.39**	0.65	**0.45**

Table 3: Transformer performance regarding learned embbeding using BPE 5000 and delexicalise format and different pretrained embeddings using lexicalise format.

6.3 Back Translation

In the context of NMT, there has been extensive work in pursuance of improving models with monolingual data. In this work, we specifically focus on BT, which operates in a semi-supervised setup where both parallel corpora and monolingual data in the target language are available (Sennrich et al., 2016).

First of all, BT trains an intermediate system on the parallel data which is used to translate the target monolingual data into the source language, i.e. text-to-RDF. The latter, results in a parallel corpus where the source, RDF, is synthetic MT output while the target is genuine text written by humans. Afterwards, the generated synthetic parallel corpus is added to the real bitext in order to train a final model that will translate from the source to the target language, equivalently RDF-to-text.

6.3.1 Monolingual Data

The datasets taken from WebNLG used in this work referred in Section 5.1 were derived from the Wikipedia database. Likewise, we decided to use english monolingual data extracted from the same source. There are many publicly available datasets, but we decided not to use them as they were very heavily preprocessed towards language modeling tasks. Instead, we considered scrapping the source using a Wikipedia extractor[5] and some adapted scripts[6] to create small to medium sized training datasets from collected data.

As a first approach, we naively used the resulting datasets as input data for our BT analysis. Alternatively, we implemented a more targeted approach to the training data since entities appearing in the corpus were annotated. Taking this into account, not only was the team capable of scrapping Wikipedia pages of most similar entities to the ones in the corpus, but we were also able to limit scrapping to the first three paragraph in each page without loss of quality. In order to determine the most similar entities an embedding distance-based approach was computed regarding Wikipedia2Vec (Yamada et al., 2020) that allows to query for entities rather than words. For instance, 'Barack Obama' most similar entities can be queried without lowercasing either splitting, which is not allowed in most pretrained embeddings.

6.3.2 Back-Translation Model

The team began considering that model's architecture used to generate text from RDF could also be trained in the reverse direction: text-to-RDF. This model was configured with the hyperparameters mentioned in the previous Section 5.2. Once trained, the synthetic data generated from both monolingual text was pretty poor since most of the predicted RDFs had nothing in common with the text.

Thus, we noticed a small difference in our task with respect to MT when applying BT, which is that MT solves the same task in both directions, as it is working in a text-to-text context. However, this is not our case were RDF-to-text is a generative

[5]https://github.com/attardi/wikiextractor

[6]https://github.com/n-waves/multifit/

BT Model	Embedding	Dimension	BPE	BLEU (↑)	METEOR (↑)	chrF++ (↑)	TER (↓)
Transformer	GloVe	300	—	31.62	0.30	0.54	0.61
Parsing	GloVe	300	—	38.26	0.35	0.61	0.51
Parsing	Learned	256	5000	**41.97**	**0.39**	**0.65**	**0.45**

Table 4: Transformer performance after training with the real and synthetic corpus, 79639 instances almost doubling the original dataset size, generated by different BT models.

task while text-to-RDF is a parsing task, hence, we adopted a different approach to solve the parsing task.

The new approach consisted of parse trees that guarantee that elements in the RDF appear in the text. We updated and adapted the script proposed here[7] that is based on the algorithm presented by (Rusu et al., 2007).

In Table 4, we report the performance of the Transformer model after training with the real data and the synthetic corpus generated from the second monolingual data, the targeted one. As we metioned, the synthetic data generated from the Transformer model was of little quality, in fact, training with this data lead to achieve the worst results yet presented. Although parsing approach significantly improve synthetic data and its performance, results were worse in comparison with training without this synthetic corpus, as in Table 3. Not until learned embeddings and BPE were considered did we notice some improvement with respect to training with only original data. Combining learned embeddings with BPE improved their best performance obtained in Table 2 by up to 3 BLEU points. This combination almost achieves same results to best GloVe pretrained embeddings, Table 5, and even surpasses them in TER metric.

Model	BLEU (↑)	METEOR (↑)	chrF++ (↑)	TER (↓)
Vanilla	32.99	0.33	0.56	0.61
BPE	38.33	0.36	0.62	0.53
Embedding	**42.85**	**0.39**	**0.66**	0.46
BT	41.97	**0.39**	0.65	**0.45**

Table 5: Performance comparison between vanilla model, Convolutional-based, and best results of each exeriment, Transformer-based.

7 Conclusions

This work pretend to solve the RDF-to-text task by means of Machine Translation techniques, which falls in a different Natural Language Generation domain, text-to-text, rather than data-to-text. The team observed that if delexicalisation step is performed with Byte Pair Encoding (BPE) and learning embeddigs, the better the number of words in BPE is tunned, the better the results can be. Moreover, we showed that learning embeddings are worthless considering in comparison to use pretrained embeddings, such as GloVe or Wikipedia2Vec. However, the performance obtained with learning embeddings and BPE significantly increased, reaching the best results presented in this work, when synthetic corpus was used in training phase, as it did not happen with pretrained embeddings. In future work, we plan to enlarge the synthetic corpus since Back Translation results are promising and monolingual text is extensively available.

[7]`https://github.com/calosh/RDF-Triple-API`

Acknowledgment

We want to specially thank José A. Rodríguez Fonollosa for his feedback and technical support during the development of this paper.

References

Angela Fan Yann Dauphin Mike Lewis. Hierarchical Neural Story Generation // arXiv. 05 2018.

Bahdanau Dzmitry, Cho Kyunghyun, Bengio Yoshua. Neural Machine Translation by Jointly Learning to Align and Translate // CoRR. 2015. abs/1409.0473.

Banerjee Satanjeev, Lavie Alon. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments // Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization. Ann Arbor, Michigan: Association for Computational Linguistics, VI 2005. 65–72.

Chung Junyoung, Gülçehre Çaglar, Cho KyungHyun, Bengio Yoshua. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling // CoRR. 2014. abs/1412.3555.

Ding Li, Finin Tim, Joshi Anupam, Pan Rong, Cost R. Scott, Peng Yun, Reddivari Pavan, Doshi Vishal, Sachs Joel. Swoogle: A Search and Metadata Engine for the Semantic Web // Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management. New York, NY, USA: Association for Computing Machinery, 2004. 652–659. (CIKM '04).

Fader Anthony, Zettlemoyer Luke, Etzioni Oren. Open Question Answering over Curated and Extracted Knowledge Bases // Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, NY, USA: Association for Computing Machinery, 2014. 1156–1165. (KDD '14).

Gardent Claire, Shimorina Anastasia, Narayan Shashi, Perez-Beltrachini Laura. Creating Training Corpora for NLG Micro-Planners // Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2017a. 179–188.

Gardent Claire, Shimorina Anastasia, Narayan Shashi, Perez-Beltrachini Laura. The WebNLG Challenge: Generating Text from RDF Data // Proceedings of the 10th International Conference on Natural Language Generation. Santiago de Compostela, Spain: Association for Computational Linguistics, IX 2017b. 124–133.

Gatt Albert, Krahmer Emiel. Survey of the State of the Art in Natural Language Generation: Core tasks, applications and evaluation // CoRR. 2017. abs/1703.09902.

Gehring Jonas, Auli Michael, Grangier David, Yarats Denis, Dauphin Yann N. Convolutional Sequence to Sequence Learning // CoRR. 2017. abs/1705.03122.

Hochreiter Sepp, Schmidhuber Jürgen. Long Short-term Memory // Neural computation. 12 1997. 9. 1735–80.

Huang Zan, Chung Wingyan, Ong Thian-Huat, Chen Hsinchun. A Graph-Based Recommender System for Digital Library // Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries. New York, NY, USA: Association for Computing Machinery, 2002. 65–73. (JCDL '02).

Kingma Diederik P., Ba Jimmy. Adam: A Method for Stochastic Optimization // CoRR. 2015. abs/1412.6980.

Koehn Philipp, Hoang Hieu, Birch Alexandra, Callison-Burch Chris, Federico Marcello, Bertoldi Nicola, Cowan Brooke, Shen Wade, Moran Christine, Zens Richard, Dyer Chris, Bojar Ondřej, Constantin Alexandra, Herbst Evan. Moses: Open Source Toolkit for Statistical Machine Translation // Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions. Prague, Czech Republic: Association for Computational Linguistics, VI 2007. 177–180.

Marcheggiani Diego, Perez-Beltrachini Laura. Deep Graph Convolutional Encoders for Structured Data to Text Generation // arXiv. 10 2018.

Mendes Pablo, Jakob Max, Bizer Christian. DBpedia: A Multilingual Cross-domain Knowledge Base // Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12). Istanbul, Turkey: European Language Resources Association (ELRA), V 2012. 1813–1817.

Micikevicius Paulius, Narang Sharan, Alben Jonah, Diamos Gregory F., Elsen Erich, García David, Ginsburg Boris, Houston Michael, Kuchaiev Oleksii, Venkatesh Ganesh, Wu Hao. Mixed Precision Training // CoRR. 2017. abs/1710.03740.

Papineni Kishore, Roukos Salim, Ward Todd, Zhu Wei-Jing. Bleu: a Method for Automatic Evaluation of Machine Translation // Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, VII 2002. 311–318.

Pascanu Razvan, Mikolov Tomas, Bengio Yoshua. Understanding the exploding gradient problem // CoRR. 2012. abs/1211.5063.

Pennington Jeffrey, Socher Richard, Manning Christopher D. GloVe: Global Vectors for Word Representation // Empirical Methods in Natural Language Processing (EMNLP). 2014. 1532–1543.

Popović Maja. chrF++: words helping character n-grams // Proceedings of the Second Conference on Machine Translation. Copenhagen, Denmark: Association for Computational Linguistics, IX 2017. 612–618.

Rusu Delia, Dali Lorand, Fortuna Blaž, Grobelnik Marko, Mladenić Dunja. Triplet extraction from sentences // Proc. 10th. Int. Multiconference Inform. Soc., Ljubljana, Slovenia. 2007. 8–12.

Sennrich Rico, Haddow Barry, Birch Alexandra. Neural Machine Translation of Rare Words with Subword Units // CoRR. 2015. abs/1508.07909.

Sennrich Rico, Haddow Barry, Birch Alexandra. Improving Neural Machine Translation Models with Monolingual Data // Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Berlin, Germany: Association for Computational Linguistics, VIII 2016. 86–96.

Shimorina Anastasia, Gardent Claire. Handling Rare Items in Data-to-Text Generation // Proceedings of the 11th International Conference on Natural Language Generation. Tilburg University, The Netherlands: Association for Computational Linguistics, XI 2018. 360–370.

Snover Matthew, Dorr Bonnie, Schwartz Richard, Micciulla Linnea, Makhoul John. A study of translation edit rate with targeted human annotation // In Proceedings of Association for Machine Translation in the Americas. 2006. 223–231.

Srivastava Nitish, Hinton Geoffrey, Krizhevsky Alex, Sutskever Ilya, Salakhutdinov Ruslan. Dropout: A Simple Way to Prevent Neural Networks from Overfitting // Journal of Machine Learning Research. 2014. 15, 56. 1929–1958.

Sutskever Ilya, Vinyals Oriol, Le Quoc V. Sequence to Sequence Learning with Neural Networks // CoRR. 2014. abs/1409.3215.

Szegedy Christian, Vanhoucke Vincent, Ioffe Sergey, Shlens Jonathon, Wojna Zbigniew. Rethinking the Inception Architecture for Computer Vision // CoRR. 2015. abs/1512.00567.

Trisedya Bayu Distiawan, Qi Jianzhong, Zhang Rui, Wang Wei. GTR-LSTM: A Triple Encoder for Sentence Generation from RDF Data // Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Melbourne, Australia: Association for Computational Linguistics, VII 2018. 1627–1637.

Vaswani Ashish, Shazeer Noam, Parmar Niki, Uszkoreit Jakob, Jones Llion, Gomez Aidan N., Kaiser Lukasz, Polosukhin Illia. Attention Is All You Need // CoRR. 2017. abs/1706.03762.

Yamada Ikuya, Asai Akari, Sakuma Jin, Shindo Hiroyuki, Takeda Hideaki, Takefuji Yoshiyasu, Matsumoto Yuji. Wikipedia2Vec: An Efficient Toolkit for Learning and Visualizing the Embeddings of Words and Entities from Wikipedia // arXiv preprint 1812.06280v3. 2020.

Zhu Yaoming, Wan Juncheng, Zhou Zhiming, Chen Liheng, Qiu Lin, Zhang Weinan, Jiang Xin, Yu Yong. Triple-to-Text: Converting RDF Triples into High-Quality Natural Languages via Optimizing an Inverse KL Divergence // CoRR. 2019. abs/1906.01965.