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

Oriol Domingo, David Bergés, Roser Cantenys, Roger Creus, José A. R. Fonollosa
Universitat Politècnica de Catalunya, Barcelona
TALP Research Center
{oriol.domingo.roig, david.bergés
roser.cantenys, roger.creus}@estudiantat.upc.edu
{jose.fonollosa}@upc.edu

Abstract

This work establishes key guidelines on how, which and when Machine Translation (MT) techniques are worth applying to RDF-to-Text task. Not only do we apply and compare the most prominent MT architecture, the Transformer, but we also analyze state-of-the-art techniques such as Byte Pair Encoding or Back Translation to demonstrate an improvement in generalization. In addition, we empirically show how to tailor these techniques to enhance models relying on learned embeddings rather than using pretrained ones. Automatic metrics suggest that Back Translation can significantly improve model performance up to 7 BLEU points, hence, opening a window for surpassing state-of-the-art results with appropriate architectures[1].

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 Framework (RDF), which consist of three elements: $\langle$*subject, predicate, object*$\rangle$. Thus, it establishes relations (*predicate*) between entities (*subject, object*).

It is known that KBs 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 most common approach taken to solve the RDF-to-Text task, since it is based on a mapping from structured data to text. However, we focus on applying 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 assess the advantages of Back-Translation (BT) (Sennrich et al., 2016) for this task, as well as, compare the benefits of learning the embeddings from scratch to the alternative of providing pretrained embeddings.

The specific contributions of our work to the field are:

- We train an encoder-decoder Transformer architecture, in end-to-end and pipeline manners, for RDF-to-Text task.
- We enhance our models with Byte Pair Encoding (BPE) (Sennrich et al., 2015), embedding analysis and BT for better generalization. To the best of our knowledge, our BT experiment is the first ever applied in the RDF-to-Text domain. In this paper is shown that working with synthetic corpus leads to best results as long as models consider BPE and learned embedding techniques in an end-to-end environment.

2 Task Formulation

The RDF-to-Text task aims to generate natural text from a set of RDF, which are entities in the form of words establishing relations between them.

The input to our system is a KB that can be denoted as a set of RDF, i.e. $\mathcal{K} := \{r_1, ..., r_n\}$. Each element in RDF r_i can be defined as $\langle s_i, p_i, o_i \rangle$, these elements stand for subject, predicate and object, respectively.

[1]Data and source code available at https://github.com/uridr/RDF-TextGeneration

3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+), Dublin, Ireland (Virtual), 18 December 2020, pages 40–47,

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

Table 1: Lexicalise and delexicalise language.

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 assess the correctness of $\mathcal{S}$, the following metrics are studied: BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), chrF++ (Popović, 2017); for which greater values mean better performance, and TER (Snover et al., 2006); for which lower values means better performance.

3 Related Work

The present task was also proposed in the WebNLG Challenge 2017[2]. Submissions to this challenge included different approaches, such as: Neural Machine Translation (NMT), Statistical Machine Translation (SMT) and Pipeline systems (Gardent et al., 2017b). The best model regarding automatic metrics was submitted by the University of Melbourne. Such model consisted of an encoder-decoder Bidirectional Long Short-Term Memory architecture with attention[3]. Although the model falls in NMT field, it is not an strict end-to-end approach as it matches input and output with delexicalised templates.

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. It can be empirically shown how encoder-decoder GNN architectures can achieve similar results to the best submission in WebNLG Challenge 2017 (Marcheggiani and Perez-Beltrachini, 2018) or even improve these benchmarks (Trisedya et al., 2018).

4 System Architecture

This section describes the pipeline architecture from preprocessing $\mathcal{K}$ to postprocessing the output of intermediate models. However, the system will be adapted to allow different experiments.

4.1 Preprocessing

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 generalised 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, BPE is also applied to improve the translation quality. More details about the advantages and disadvantages of this technique will be discused in Section 6.1.

4.2 Transformer model

The team implemented a sequence-to-sequence, encoder-decoder based on the Transformer model proposed in (Vaswani et al., 2017). Moreover, Transformer can be interpreted as a special case of GNN, mentioned by Chaitanya Joshi[4]. Hence, this approach is also aligned with the latest research, mainly based on GNN, conducted to solve the RDF-to-text task.

This model is implemented[5] 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 tokenization as well as BPE. Moreover, we need to perform a relexicalisation step in order to recover the specific relationships and meanings from the original lexicalise RDF.

[2] https://webnlg-challenge.loria.fr

[3] https://webnlg-challenge.loria.fr/files/melbourne_report.pdf

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

[5] See Appendix A for additional details on the hyperparam-

Embedding	Dimension	Format	BPE	BLEU (↑)	METEOR (↑)	chrF++ (↑)	TER (↓)
Learned	256	Delexicalise	5000	38.33	0.36	0.62	0.53
GloVe	100	Lexicalise	—	36.98	0.35	0.60	0.48
GloVe	200	Lexicalise	—	42.41	**0.39**	**0.66**	0.46
GloVe	300	Lexicalise	—	**42.85**	**0.39**	**0.66**	0.46
Wikipedia2Vec	100	Lexicalise	—	37.45	0.35	0.61	0.48
Wikipedia2Vec	300	Lexicalise	—	41.69	**0.39**	0.65	**0.45**

Table 2: Transformer performance with learned embbeding using BPE 5000 and delexicalise format, and different pretrained embeddings using lexicalise format in release_v2.1 dataset. Not considering BPE is marked as —.

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 3: Transformer performance regarding different number of BPE subwords in release_v2.1 dataset. All models are trained with delexicalisation and learned embeddings of 256-dimensions. Not considering BPE is top-row, marked as —.

5 Data

The data used in this work is the release_v2.1 and webnlg_challenge_2017 (Gardent et al., 2017a) taken from the WebNLG corpus[6]. Both datasets are partitioned into three subsets: train, dev and test. The release_v2.1 respectively have 34338, 4313 and 4222 instances, and the webnlg_challenge_2017 have 18102, 2268 and 1862 with unseen relations in the test set. These datasets are based on DBPedia, which is a multilingual KB that was built from several kinds of structured information included in Wikipedia (Mendes et al., 2012).

6 Ablation Experiment and Results

The first experiment analyses the influence of the number of BPE subwords. The second experiment study the improvements of pretrained embeddings against learned embeddings. Last experiment is designed to enlarge training data by means of different BT approaches.

6.1 Byte Pair Encoding

It has been demonstrated that the method used to treat rare items in data-to-text generation strongly impacts system performance (Shimorina and Gardent, 2018). The authors (Shimorina and Gardent, 2018) also stated that character-based techniques, such as BPE, obtained very poor results in the WebNLG dataset. Nevertheless, the combination of delexicalisation and BPE was 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 3 shows that not considering BPE 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 more robust to data that consist of unseen vocabulary than systems that do not consider such a technique since they would decrease their performance as unknown vocabulary increases, attaining greater differences.

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. We found that suitable embeddings could be: GloVe (Pennington et al., 2014) and Wikipedia2Vec (Yamada et al., 2020) since instances are extracted from Wikipedia.

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 the delexicalisation step was giving, it is no longer present using these pretrained embeddings. However, these embeddings have been built using a large amount

eter tuning and optimizing approaches.

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

BT Model	Tagged	Corpus	Embedding	Dimension	BPE	BLEU (↑)	METEOR (↑)	chrF++ (↑)	TER (↓)
Transformer	No	79639	GloVe	300	—	31.62	0.30	0.54	0.61
Parsing	No	79639	GloVe	300	—	38.26	0.35	0.61	0.51
Parsing	No	79639	Learned	256	5000	41.97	0.39	0.65	0.45
Parsing	Yes	79639	Learned	256	5000	43.37	**0.40**	**0.67**	**0.43**
Parsing	No	155897	Learned	256	10000	44.01	**0.40**	**0.67**	0.44
Parsing	Yes	155897	Learned	256	10000	**44.22**	**0.40**	**0.67**	**0.43**

Table 4: Transformer performance after training with the release_v2.1 and synthetic corpus in BT experiment.

of data that should lead to good generalization between entities as well.

In Table 2, it is clearly shown how the learned embedding approach with BPE can be surpassed regarding any metric even with lower dimensionality, 200-dimension GloVe, by the pretrained embedding approach. Nevertheless, we will see in the following Section 6.3 that adding more instances benefits more the learned embedding approach rather than the pretrained one. This might reveal that the learned embedding approach is underperforming the pretrained embedding approach at this stage due to a low number of data instances.

Finally, there is not a notable difference between 200-dimension GloVe, 300-dimension GloVe and 300-dimension Wikipedia2Vec.

6.3 Back-Translation

This work specifically focuses 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.

Moreover, we also studied the performance of Tagged Back-Translation (Caswell et al., 2019). This technique adds an extra token at the beginning of each synthetic instance allowing the model to differentiate them from real data.

6.3.1 Monolingual Data

We used english monolingual data extracted from Wikipedia. The scrapped pages were from most similar entities to the ones in the training corpus following an embedding distance-based approach. This distance 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

There is 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 task while text-to-RDF is a parsing task. Thus, our best 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 model after training with real and synthetic data. The synthetic data generated from the Transformer 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 2. 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 3 by almost 6 BLEU points, 0.04 METEOR points, 0.05 chrF++ points and 0.1 TER points. Furthermore, Tagged BT obtained similar or better metrics even when training was done with nearly half of the data used in the best BT output.

7 Models Summary

In the following, the best models obtained in each of the considered experiments as well as their performance are presented. Moreover, we also provide a comparison between our models and some of the most relevant models in the RDF-to-Text domain.

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

WebNLG	⟨Alfred Moore Scales, successor, Daniel Gould Fowle ⟩, ⟨ Daniel Gould Fowle, alma mater, Princeton University⟩
BPE	The alma mater of Alfred Moore Scales was the Princeton University and he was succeeded by Daniel Gould Fowle.
Embedding	Daniel Gould Fowle succeeded Alfred Moore Scales , whose alma mater was the university of Gottingen.
BT	Alfred Moore Scales was succeeded by Daniel Gould Fowle and his alma mater was Princeton University.
Tagged BT	Daniel Gould Fowle succeeded Alfred Moore Scales.

Table 5: Example of a set of RDF from the test set, top, and the prediction of each model in different experiments.

Model	**BLEU** (↑)	**METEOR** (↑)	**chrF++** (↑)	**TER** (↓)
BPE	38.33	0.36	0.62	0.53
Embedding	42.85	0.39	0.66	0.46
BT	44.01	**0.40**	**0.67**	0.44
Tagged BT	**44.22**	**0.40**	**0.67**	**0.43**

Table 6: Performance summary between best results of each experiment using a Transformer architecture in release_v2.1 dataset.

Model	**BLEU** (↑)		**METEOR** (↑)		**TER** (↓)	
	Seen	Unseen	Seen	Unseen	Seen	Unseen
Transformer	40.45	4.56	0.38	0.11	0.48	1.06
BT Transformer	39.9	21.44	0.37	0.29	0.46	0.73
Melbourne (3)	54.52	33.27	**0.41**	**0.33**	**0.37**	**0.55**
(Trisedya et al., 2018)	**58.6**	**34.1**	0.4	0.32	0.41	0.57

Table 7: Comparison between Transformer with BPE 1000 and learned embeddings, BT Transformer with BPE 10000 and learned embedding, and state-of-the-art models with respect to webnlg_challenge_2017 dataset.

7.1 Work Comparison

Models considering pretrained embeddings rather than learned embeddings can lead to good performance metrics, however, BT and Tagged BT are better when embeddings are learned from scratch. Not only was the best performance achieved with BT, see Table 6, but the text generated from BT models were more coherent with the inputs as shown in Table 5, where both BT models are the ones to only include relations appearing in the input.

7.2 Related Work Comparison

Table 7 depicts a comparative analysis between Transformer architecture trained end-to-end with and without synthetic data, i.e. considering BT, and state-of-the-art models, both presented in Section 3. It is clearly shown how the performance in unseen domain is improved when considering BT in the Transformer. In fact, there is a significant metric improvement when BT is applied (i.e. 7 BLEU point improvement). Notice also that the performance in the seen domain is quite invariant to the improvement in the unseen domain. Hence, state-of-the-art models could still perform much better with BT.

This finding opens a window for training these state-of-the-art models with our synthetic corpus, if possible. Expecting these results to be very prominent due to the fact that we have demonstrated that training with synthetic data in an end-to-end manner is worth applying in RDF-to-Text domain.

8 Conclusions

This work provides a solution to the RDF-to-text task by means of MT techniques, which falls in a different NLG domain, text-to-text, rather than data-to-text. The team observed that if a delexicalisation step is performed with BPE and learning embeddigs, the better the number of words in BPE is tunned, the better the results can be. Moreover, we showed that pretrained embeddings are worth considering in comparison to use learned embeddings. The performance obtained with learning embeddings and BPE significantly increased ($\Delta \simeq 7$ BLEU, $\Delta \simeq 0.05$ METEOR, $\Delta \simeq 0.05$ chrF++ and $\nabla \simeq 0.1$ TER), reaching the best results presented in this work, when synthetic corpus was used in training phase, as it did not happen with pretrained embeddings. Hence, revealing that BPE and learned embeddings benefit more than pretrained embeddings from a larger dataset. In future work, we plan to enlarge the synthetic corpus since Back Translation results are promising and monolingual text is extensively available, and use this synthetic corpus to train other relevant models in the RDF-to-Text task.

Acknowledgment

We want to especially thank Paula M. Petrone for her enriching and constructive comments on the paper, and anonymous reviewers for their thoughtful comments on the paper as well. This work was supported by the project ADAVOICE, PID2019-107579RB-I00 / AEI / 10.13039/501100011033.

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Isaac Caswell, Ciprian Chelba, and David Grangier. 2019. Tagged back-translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pages 53–63, Florence, Italy. Association for Computational Linguistics.

Li Ding, Tim Finin, Anupam Joshi, Rong Pan, R. Scott Cost, Yun Peng, Pavan Reddivari, Vishal Doshi, and Joel Sachs. 2004. Swoogle: A search and metadata engine for the semantic web. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, CIKM '04, page 652–659, New York, NY, USA. Association for Computing Machinery.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, page 1156–1165, New York, NY, USA. Association for Computing Machinery.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017a. Creating training corpora for nlg micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 179–188. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017b. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133, Santiago de Compostela, Spain. Association for Computational Linguistics.

Albert Gatt and Emiel Krahmer. 2017. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *CoRR*, abs/1703.09902.

Zan Huang, Wingyan Chung, Thian-Huat Ong, and Hsinchun Chen. 2002. A graph-based recommender system for digital library. In *Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '02, page 65–73, New York, NY, USA. Association for Computing Machinery.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Diego Marcheggiani and Laura Perez-Beltrachini. 2018. Deep graph convolutional encoders for structured data to text generation. *arXiv*.

Pablo Mendes, Max Jakob, and Christian Bizer. 2012. DBpedia: A multilingual cross-domain knowledge base. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 1813–1817, Istanbul, Turkey. European Language Resources Association (ELRA).

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

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

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

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

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

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages

86–96, Berlin, Germany. Association for Computational Linguistics.

Anastasia Shimorina and Claire Gardent. 2018. Handling rare items in data-to-text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 360–370, Tilburg University, The Netherlands. Association for Computational Linguistics.

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

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2015. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567.

Bayu Distiawan Trisedya, Jianzhong Qi, Rui Zhang, and Wei Wang. 2018. GTR-LSTM: A triple encoder for sentence generation from RDF data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1627–1637, Melbourne, Australia. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Ikuya Yamada, Akari Asai, Jin Sakuma, Hiroyuki Shindo, Hideaki Takeda, Yoshiyasu Takefuji, and Yuji Matsumoto. 2020. Wikipedia2vec: An efficient toolkit for learning and visualizing the embeddings of words and entities from wikipedia. *arXiv preprint 1812.06280v3*.

A Appendix

This appendix describes the training regime for the proposed models.

A.1 Model Parameters and Optimization

We used a Transformer model with a total of 3 layers with Feed Forward Networks of dimensionality 1024 and 8 attention heads, performing cross + self attention at each layer. We used 256 dimension embeddings, shared across the entire network plus fixed (not learned) sinusoidal positional encodings.

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.

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.

A.1.1 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.

A.2 Optimizer

We used the Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. We increased the learning rate linearly for a total of 4000 warming steps to $1e^{-03}$, and decreased it following an inverse square root formula from there. Additionally, we applied several regularization techniques such as dropout, gradient clipping and label smoothing for our loss formula.

$$lr_i = \frac{lr_0 \cdot \sqrt{warmup\_updates}}{\sqrt{i}} \quad (1)$$

A.2.1 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 sub-layer 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 renormalised the gradients if their norm exceeded 0.1 (Pascanu et al., 2012).

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.