Joint translation and unit conversion for end-to-end localization

Georgiana Dinu Prashant Mathur Marcello Federico Stanislas Lauly Yaser Al-Onaizan

Amazon AWS AI

{gddinu,pramathu,marcfede,laulysl,onaizan}@amazon.com

Abstract

A variety of natural language tasks require processing of textual data which contains a mix of natural language and formal languages such as mathematical expressions. In this paper, we take unit conversions as an example and propose a data augmentation technique which lead to models learning both translation and conversion tasks as well as how to adequately switch between them for end-to-end localization.

1 Introduction

Neural networks trained on large amounts of data have been shown to achieve state-of-the art solutions on most NLP tasks such as textual entailment, question answering, translation, etc. In particular, these solutions show that one can successfully model the ambiguity of language by making very few assumptions about its structure and by avoiding any formalization of language. However, unambiguous, formal languages such as numbers, mathematical expressions or even programming languages (e.g. markup) are abundant in text and require the ability to model the symbolic, "procedural" behaviour governing them. (Ravichander et al., 2019; Dua et al., 2019).

An example of an application where such examples are frequent is the extension of machine translation to localization. Localization is the task of combining translation with "culture adaptation", which involves, for instance, adapting dates (*12/21/2004* to *21.12.2004)*, calendar conversions (*March 30, 2019* to *Rajab 23, 1441* in Hijri Calendar) or conversions of currencies or of units of measure (*10 kgs* to *22 pounds*).

Current approaches in machine translation handle the processing of such sub-languages in one of two ways: The sub-language does not receive any special treatment but it may be learned jointly with the main task if it is represented enough in the data. Alternatively, the sub-language is decoupled from the natural text through pre/post processing techniques: e.g. a *miles* expression is converted into *kilometers* in a separate step after translation.

Arguably the first approach can successfully deal with some of these phenomena: e.g. a neural network may learn to invoke a simple conversion rule for dates, if enough examples are seen training. However, at the other end of the spectrum, correctly converting distance units, which itself is a simple algorithm, requires knowledge of numbers, basic arithmetic and the specific conversion function to apply. It is unrealistic to assume a model could learn such conversions from limited amounts of parallel running text alone. Furthermore, this is an unrealistic task even for distributional, unsupervised pre-training (Turney and Pantel, 2010; Baroni and Lenci, 2010; Peters et al., 2018), despite the success of such methods in capturing other non-linguistic phenomena such as world knowledge or cultural biases (Bolukbasi et al., 2016; Vanmassenhove et al., 2018).[1]

While the second approach is currently the preferred one in translation technology, such decoupling methods do not bring us closer to end-to-end solutions and they ignore the often tight interplay of the two types of language: taking unit conversion as an example, *approximately 500 miles*, should be translated into *ungefähr 800 km* (approx. 800km) and not *ungefähr 804 km* (approx. 804km).

In this paper we highlight several of such language mixing phenomena related to the task of localization for translation and focus on two distance (miles to kilometers) and temperature (Fahrenheit to Celsius) conversion tasks. Specifically, we per-

[1] (Wallace et al., 2019) show that numeracy is encoded in pre-trained embeddings. While promising, this does not show that more complex and varied manipulation of numerical expressions can be learned in a solely unsupervised fashion.

Proceedings of the 17th International Conference on Spoken Language Translation (IWSLT), pages 265–271

https://doi.org/10.18653/v1/P17

form experiments using the popular MT transformer architecture and show that the model is successful at learning these functions from symbolically represented examples. Furthermore, we show that data augmentation techniques together with small changes in the input representation produce models which can both translate and appropriately convert units of measure in context.

2 Related work

Several theoretical and empirical works have addressed the computational capabilities end expressiveness of deep learning models. Theoretical studies on language modeling have mostly targeted simple grammars from the Chomsky hierarchy. In particular, Hahn (2019) proves that Transformer networks suffer limitations in modeling regular periodic languages (such as $a^n b^n$) as well as hierarchical (context-free) structures, unless their depth or self-attention heads increase with the input length. On the other hand, Merrill (2019) proves that LSTM networks can recognize a subset of periodic languages. Also experimental papers analyzed the capability of LSTMs to recognize these two language classes (Weiss et al., 2018; Suzgun et al., 2019; Sennhauser and Berwick, 2018; Skachkova et al., 2018; Bernardy, 2018), as well as natural language hierarchical structures (Linzen et al., 2016; Gulordava et al., 2018). It is worth noticing, however, that differently from formal language recognition tasks, state of the art machine translation systems (Barrault et al., 2019; Niehues et al., 2019) are still based on the Transformer architecture .

Other related work addresses specialized neural architectures capable to process and reason with numerical expressions for binary addition, evaluating arithmetic expressions or other number manipulation tasks (Joulin and Mikolov, 2015; Saxton et al., 2019; Trask et al., 2018; Chen et al., 2018). While this line of work is very relevant, we focus on the natural intersection of formal and everyday language. The types of generalization that these studies address, such as testing with numbers orders of magnitude larger than those in seen in training, are less relevant to our task.

The task of solving verbal math problems (Mitra and Baral, 2016; Wang et al., 2017; Koncel-Kedziorski et al., 2016; Saxton et al., 2019) specifically addresses natural language mixed with formal language. Similarly, (Ravichander et al., 2019) introduces a benchmark for evaluating quantitative reasoning in natural language inference and (Dua et al., 2019) one for symbolic operations such as addition or sorting in reading comprehension. However these papers show the best results with two-step approaches, which extract the mathematical or symbolic information from the text and further manipulate it analytically. We are not aware of any other work successfully addressing both machine translation and mathematical problems, or any of the benchmarks above, in an end-to-end fashion.

3 Unit conversion in MT localization

The goal of localization is to enhance plain content translation so that the final result looks and feels as being created for a specific target audience.

Parallel corpora in general include localization of formats numeric expressions (e.g. from *1,000,000.00* (en-us) to *1.000.000,00* (de-de)). Format conversions in most of the cases reduce to operations such as reordering of elements and replacement of symbols, which quite naturally fit inside the general task of machine translation. In this paper, we are interested in evaluating the capability of neural MT models to learn less natural operations, which are typically involved in the conversion of time expressions (e.g. 3:30pm → 15:30) and units of measure, such as lengths (10ft to 3m) and temperatures (55F to 12.8C).

We choose two measure unit conversion tasks that are very prevalent in localization: Fahrenheit to Celsius temperature conversion and miles to kilometers. We address the following questions: 1) Can a standard NMT architecture, the transformer, be used to learn the functions associated with these two conversion tasks (Section 3.1) and 2) Can the same architecture be used to train a model that can do both MT and unit conversion? (Section 3.2)

3.1 Unit conversion

Network architecture We use the state-of-the-art transformer architecture (Vaswani et al., 2017) and the Sockeye Toolkit (Hieber et al., 2017) to train a network with 4 encoder layers and 2 decoder layers for a maximum of 3000 epochs (See Appendix A for details). As the vocabulary size is small the training is still very efficient. For the experiments training several tasks jointly we facilitate the context-switching between the different tasks with an additional token-level parallel stream (source factors) (Sennrich and Haddow, 2016). We use two values for the digits in numerical expres-

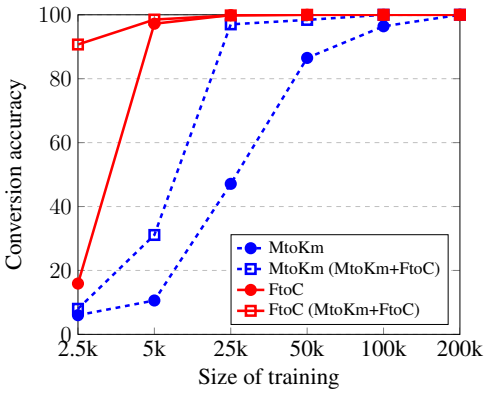

Figure 1: Conversion accuracy with $\pm 10^{-4}$ tolerance on relative error, as a function of the number of the target conversion examples in the train data. Functions are learned both in isolation and in a joint setting (MtoKm + FtoC) which adds to training an equal amount of data for the other function.

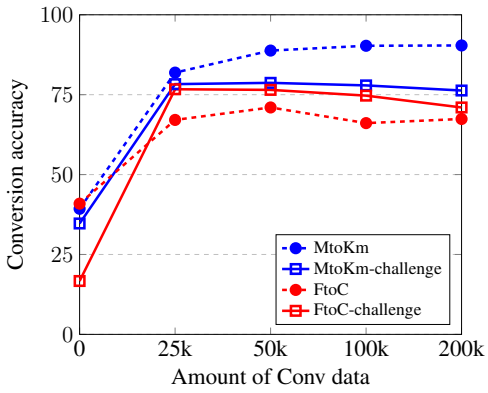

Figure 2: Accuracy of localization conversion (tolerance 0.01%) on regular and *challenge* sets. All models use source factors and are trained using: 2.2M MT data + 15k Loc data + varying amounts of Conv data.

sions (distance/temperature) and a third value for all other tokens. These are concatenated to each token as 8-dimensional embeddings.

Data The models are trained with parallel examples of the two functions, one affine: $°\text{F} \rightarrow °\text{C}(x) = (x - 32) \times \frac{5}{9}$ and one linear: $\text{mi} \rightarrow \text{km}(x) = x \times 1.60934$. For each task, we generate training data of various input lengths ranging from 1 to 6 digits in the input. The input is distributed uniformly w.r.t 1) integer versus single digit precision (with the output truncated to same precision as the input) and 2) the length of the input in digits. We over-sample when there are not enough distinct data points, such as in the case of double- or single-digit numbers. The numerical input is tokenized into digits (e.g. *5 2 1 miles*) and we train individual models for the two functions, as well as joint models, using held-out data for validation and testing. Note that unlike previous work, we are interested only in interpolation generalization: test numbers are unseen, but the *range* of test numbers does not increase.

Results Results as a function of the amount of training data are given in Figure 1. Test sets are synthetic and contain numbers in $[10^3 - 10^6]$ range.

The results show that the transformer architecture can learn the two functions perfectly, however, interestingly enough, the two functions are learned differently. While the degree conversion is learned with a high accuracy with as little as several thousand examples, the distance conversion is learned gradually, with more data leading to better and better numerical approximations: in this case the model reaches high precision in conversion only with data two orders of magnitude larger. Both functions are learned with less data when training is done jointly and source factors are used - this suggests that, despite the fact that the functions are very different, joint training may facilitate the learning of numbers as a general concept and helps learn additional functions more efficiently.

3.2 Joint MT and unit conversion

In a second set of experiments we investigate if the transformer model is able to perform both the translation and the unit conversion tasks and learns to adequately switch from one to the other in context. We use the same architecture as in the previous section, with minor modifications: we use subword embeddings with a shared vocabulary of size 32000 and a maximum number of epochs of 30.

Data As standard MT parallel data we use a collection containing Europarl (Koehn, 2005) and news commentary data from WMT En→De shared task 2019 totalling 2.2 million sentences.[2] Standard translation test sets do not have, however, enough examples of unit conversions and in fact corpora such as CommonCrawl show inconsistent treatment of units. For this reason, we create a unit conversion (Localization) data set. We extract sentences containing Fahrenheit/Celsius and miles/km from a mix of open source data sets namely, ParaCrawl, DGT (Translation Memories), Wikipedia and OpenSubtitles, TED talks from OPUS (Tiedemann, 2012). Regular expressions are used to extract the sentences containing the units and modify the source or the reference by

[2]We opt for a smaller experiment in order to speed up computations and to prioritize efficiency in our experiments (Strubell et al., 2019). We have no reason to assume any dependency on the data size.

	Example	
Conv	5 2 1 miles	8 3 9 km
MT	We do not know what is happening.	Wir wissen nicht, was passiert.
Loc.	The venue is within 3 . 8 miles from the city center	Die Unterkunft ist 6 km vom Stadtzentrum entfernt

Table 1: The three types of data used in training the joint model: unit conversion data, standard MT data and localization (Loc) data containing unit conversions in context.

		news17	Loc-dist		Loc-temp	
S.f.	#Loc	Bleu	Bleu	Acc.	Bleu	Acc.
-	0	22.7	20.6	0%	16.1	0%
-	5k	22.7	56.7	52.3%	44.1	48.3%
-	15k	23.0	61.7	76.2%	48.5	80.3%
-	30k	23.0	65.0	90.3%	48.9	81.3%
✓	0	22.9	19.5	1%	16.6	3.4%
✓	5k	22.9	58.7	69.4%	46.8	64.8%
✓	15k	23.2	63.0	88.0%	48.6	77.8%
✓	30k	22.6	64.0	88.3%	48.8	79.4%

Table 2: Bleu scores and accuracy on conversion of degrees (temp) and miles (dist) expressions in Loc test sets. Conversion accuracy is computed with a tolerance of 0.01%. All models are trained using: 2.2M MT+ 100k Conv + #Loc data (col 2) for each function, with and without **S**ource **f**actors (column 1).

converting the matched units. For example, if *5 km* is matched in the reference, we modify the source expression to *3.1 miles*.[3] We are able to extract a total of 7k examples for each of the two conversion tasks and use 5k for training and 2k for testing, making sure the train/test numerical expressions are distinct.

Results In the experimental setting, we distinguish the following three types of data: translation (**MT**), conversion (**Conv**) and localization data (conversion in context) (**Loc**), and measure performance when varying amounts of Conv and Loc are used in training. Examples of these data types are given in Table 1. The first set of experiments (Table 2) uses MT and Conv data and tests the models' performance with varying amounts of Loc data. We observe that for localization performance, Loc data in training is crucial: accuracy jumps from 2% when no Loc data is used to 66% for 5k Loc and to 82%, on average, with 15k localization examples for each function (w. source factors). However, the 15k data points are obtained by up-sampling the linguistic context and replacing the unit conversions with new unit conversions, and therefore no "real" new data is added. We observe no further improvements when more Loc data is added. Regarding the use of source factors, they help when the localization data is non-existent or very limited, however their benefits are smaller otherwise.

The Bleu scores measured on a news data set as well as on the localization data sets show no degradation from a baseline setting, indicating that the additional data does not affect translation quality. The exception is the #Loc-0 setting, in which the model wrongly learns to end all localization sentences with *km* and *C* tokens respectively, as seen in the Conv data. Similarly to the previous results, temp conversions are learned either correctly or not at all while the distance ones show numerical approximation errors: When measuring exact match in conversion (0.0 tolerance), the temperature accuracy remains largely the same while the distance accuracy drops by up to 30%.

Given the observation that Loc data is crucial, we perform another set of experiments to investigate if the Conv data is needed at all. Results are shown in Figure 2. In light of the limited amount of real distinct conversions that we see in testing, we create two additional challenge sets which use the same linguistic data and replace the original conversions with additional ones uniformly distributed w.r.t the length in digits from 1 to 6. The results indicate that conversion data is equally critical, and that the conversion cannot be learned from the localization data provided alone. The localization data rather acts as a "bridge" allowing the network to combine the two tasks it has learned independently.

4 Conclusions

We have outlined natural/formal language mixing phenomena in the context of end-to-end localization for MT and have proposed a data augmentation method for learning unit conversions in context. Surprisingly, the results show not only that a single architecture can learn both translation and unit conversions, but can also appropriately switch between them when a small amount of localization data is used in training. For future work we plan to create a diverse localization test suite and investigate if implicit learning of low-level concepts such as natural numbers takes place and if unsupervised pre-training facilitates such learning.

[3] Scripts to create this data will be released, however the data used itself does not grant us re-distribution rights.

References

Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *American Journal of Computational Linguistics*, 36(4):673–721.

Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu Pal, Matt Post, and Marcos Zampieri. 2019. Findings of the 2019 conference on machine translation (WMT19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 1–61, Florence, Italy. Association for Computational Linguistics.

Jean-Philippe Bernardy. 2018. Can Recurrent Neural Networks Learn Nested Recursion. In *Linguistic Issues in Language Technology*, volume 16.

Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. *CoRR*, abs/1607.06520.

Kaiyu Chen, Yihan Dong, Xipeng Qiu, and Zitian Chen. 2018. Neural arithmetic expression calculator. *CoRR*, abs/1809.08590.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378, Minneapolis, Minnesota. Association for Computational Linguistics.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless Green Recurrent Networks Dream Hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1195–1205, New Orleans, Louisiana. Association for Computational Linguistics.

Michael Hahn. 2019. Theoretical Limitations of Self-Attention in Neural Sequence Models. *arXiv:1906.06755 [cs]*. ArXiv: 1906.06755.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2017. Sockeye: A toolkit for neural machine translation. *CoRR*, abs/1712.05690.

Armand Joulin and Tomas Mikolov. 2015. Inferring algorithmic patterns with stack-augmented recurrent nets. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 190–198, Cambridge, MA, USA. MIT Press.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand. AAMT, AAMT.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. MAWPS: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157, San Diego, California. Association for Computational Linguistics.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

William Merrill. 2019. Sequential neural networks as automata. In *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*, pages 1–13, Florence. Association for Computational Linguistics.

Arindam Mitra and Chitta Baral. 2016. Learning to use formulas to solve simple arithmetic problems. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2144–2153, Berlin, Germany. Association for Computational Linguistics.

Jan Niehues, Cattoni Roldano, Sebastian Stüker, Matteo Negri, Marco Turchi, Elizabeth Salesky, R. Sanabria, Loïc Barrault, Lucia Specia, and Marcello Federico. 2019. The iwslt 2019 evaluation campaign. In *Proceedings of IWSLT*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Abhilasha Ravichander, Aakanksha Naik, Carolyn Penstein Rosé, and Eduard H. Hovy. 2019. EQUATE: A benchmark evaluation framework for quantitative reasoning in natural language inference. *CoRR*, abs/1901.03735.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models. *CoRR*, abs/1904.01557.

Luzi Sennhauser and Robert Berwick. 2018. Evaluating the Ability of LSTMs to Learn Context-Free Grammars. In *Proceedings of the 2018 EMNLP*

Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, pages 115–124, Brussels, Belgium. Association for Computational Linguistics.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers*, pages 83–91, Berlin, Germany. Association for Computational Linguistics.

Natalia Skachkova, Thomas Trost, and Dietrich Klakow. 2018. Closing Brackets with Recurrent Neural Networks. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 232–239, Brussels, Belgium. Association for Computational Linguistics.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Mirac Suzgun, Yonatan Belinkov, and Stuart M. Shieber. 2019. On Evaluating the Generalization of LSTM Models in Formal Languages. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2019*, pages 277–286.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

Andrew Trask, Felix Hill, Scott E Reed, Jack Rae, Chris Dyer, and Phil Blunsom. 2018. Neural arithmetic logic units. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 8035–8044. Curran Associates, Inc.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *J. Artif. Int. Res.*, 37(1):141–188.

Eva Vanmassenhove, Christian Hardmeier, and Andy Way. 2018. Getting gender right in neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3003–3008, Brussels, Belgium. 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. In *NIPS*, pages 6000–6010.

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 2019. Do NLP models know numbers? probing numeracy in embeddings. 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)*, pages 5306–5314, Hong Kong, China. Association for Computational Linguistics.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854, Copenhagen, Denmark. Association for Computational Linguistics.

Gail Weiss, Yoav Goldberg, and Eran Yahav. 2018. On the Practical Computational Power of Finite Precision RNNs for Language Recognition. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 740–745, Melbourne, Australia. Association for Computational Linguistics.

A Appendix

```
encoder-config:
  act_type: relu
  attention_heads: 8
  conv_config: null
  dropout_act: 0.1
  dropout_attention: 0.1
  dropout_prepost: 0.1
  dtype: float32
  feed_forward_num_hidden: 2048
  lhuc: false
  max_seq_len_source: 101
  max_seq_len_target: 101
  model_size: 512
  num_layers: 4
  positional_embedding_type:
     fixed
  postprocess_sequence: dr
  preprocess_sequence: n
  use_lhuc: false

decoder config:
  act_type: relu
  attention_heads: 8
  conv_config: null
  dropout_act: 0.1
  dropout_attention: 0.1
  dropout_prepost: 0.1
  dtype: float32
  feed_forward_num_hidden: 2048
  max_seq_len_source: 101
  max_seq_len_target: 101
  model_size: 512
  num_layers: 2
  positional_embedding_type:
     fixed
  postprocess_sequence: dr
  preprocess_sequence: n

config_loss: !LossConfig
  label_smoothing: 0.1
  name: cross-entropy
  normalization_type: valid
  vocab_size: 32302

config_embed_source: !
   EmbeddingConfig
  dropout: 0.0
  dtype: float32
  factor_configs: null
  num_embed: 512
  num_factors: 1
  vocab_size: 32302

config_embed_target: !
   EmbeddingConfig
  dropout: 0.0
  dtype: float32
  factor_configs: null
  num_embed: 512
  num_factors: 1
  vocab_size: 32302
```