Let's be Humorous: Knowledge Enhanced Humor Generation

Hang Zhang, Dayiheng Liu, Jiancheng Lv [*], Cheng Luo
College of Computer Science, Sichuan University
zhanghang.scu@gmail.com
losinuris@gmail.com
lvjiancheng@scu.edu.cn
wulaoshi_luocheng@foxmail.com

Abstract

The generation of humor is an under-explored and challenging problem. Previous works mainly utilize templates or replace phrases to generate humor. However, few works focus on freer forms and the background knowledge of humor. The linguistic theory of humor defines the structure of a humor sentence as set-up and punchline. In this paper, we explore how to generate a punchline given the set-up with the relevant knowledge. We propose a framework that can fuse the knowledge to end-to-end models. To our knowledge, this is the first attempt to generate punchlines with knowledge enhanced model. Furthermore, we create the first humor-knowledge dataset. The experimental results demonstrate that our method can make use of knowledge to generate fluent, funny punchlines, which outperforms several baselines. Our data and code are publicly available at https://github.com/onedoge/Knowledge-Enhanced-Humor-Generation.

1 Introduction

In daily communication, humor has a significant meaning and is the high-level development of human knowledge, emotion, and expression. However, the automated generation of humor has always been a great challenge, which requires not only a deep understanding of the semantic but also a full consideration of cultural background.

Jokes are the primary carrier of humor. According to the Inconsistency Theory, a joke generally consists of set-up and punchline (Bright, 1992). Consider the example in Fig. 1: " What is Hitler's favorite sport? Ethnic cleansing and war.". The question, which is also the set-up, provides the context for this joke. The punchline, "Ethnic cleansing and war.", is usually at the end of a joke and produces a laugh.

Set-up : What is Hitler's favorite sport?
Punchline: Ethnic cleansing and war.

Knowledge triples :
(Adolf Hitler, movement, ethnic cleansing);
(Adolf Hitler, has effect, World War II);
(Adolf Hitler, position held, Reichskanzler) ;
(Adolf Hitler, movement, antisemitism) ...

Figure 1: An illustration of a satirical joke. The set-up sentence provides the context and the punchline produces a laugh. The knowledge is organized in tripletypes which is helpful to understand this joke.

Over the years, various researchers have worked on humor generation. Previous methods are mainly based on fixed templates or lexical substitution (Petrović and Matthews, 2013; Valitutti et al., 2013; Hossain et al., 2017; Yu et al., 2018). Due to lack of context, they can only produce generic and isolated jokes. Besides, background knowledge is crucial in understanding and generating jokes. In the above example, if we don't know the background of Hitler, we wouldn't feel the humor from this joke. However, as far as we know, the background knowledge of jokes has not been introduced in the current computational humor research.

As mentioned above, we propose the task of generating punchlines with the set-up and relevant knowledge. For this task, we create the first dataset that contains set-ups, punchlines and background knowledge. Furthermore, we propose a framework as shown in Fig. 3. The relative background is converted into a knowledge graph and encoded by our proposed knowledge encoder. When generating the punchline, the decoder will first attend to the information from the set-up encoder, then fuse knowledge representation by knowledge fusion layer. The experiments indicate that our model performs better than strong baselines and can generate funny punchlines.

[*] Correspondence to Jiancheng Lv.

Our contributions are threefold: (1) We make the first attempt to generate punchline with the set-up and relevant knowledge. (2) We propose a framework to integrate external knowledge into end-to-end generation framework. (3) We provide the first dataset of knowledge paired with jokes for further study.

2 Related Work

Humor Theory Incongruity theory has an essential guiding position in the field of computational humor (Binsted et al., 2006). It believes that the inconsistency between the reader's expectation and the ending of one story is the key to humor generation. On this basis, SSTH (Script-based Semantic Theory of Humor) theory is proposed (Raskin, 2012). SSTH defines the structure of a joke as set-up and punchline. The set-up provides humorous context information, including multiple possible explanations (scripts). The punchline, usually at the end of a joke, points to a surprising explanation that triggers the humorous effect. According to this theory, we explore how to generate a punchline given the set-up sentence.

Humor Generation Petrović and Matthews (2013) attempt to fill in the blank of the fixed template "I like my **X**, like I like my **Y**, **Z** " in an unsupervised way with four customized hypotheses. Valitutti et al. (2013) substitute words with taboo words to generate adult jokes. Hossain et al. (2017) use a classifier to help people choose humorous words in a fill-in-the-blank (Liu et al., 2019) game. Yu et al. (2018) encode multiple meanings of a word and use a hybrid beam search method to generate puns. He et al. (2019) propose a retrieve-and-edit pipeline to generate a pun sentence. Different from these work, we consider the punchline generation with the use of world knowledge.

3 Humor Corpus with Background Knowledge

To prepare our dataset, we choose Short Jokes dataset[1] and Reddit-Joke dataset[2] as raw data, which are public on Kaggle. Then we perform joke filtering, punchline segmentation, and joke de-duplication. We first remove the data that contains the special characters and only keep the jokes with at least two sentences and fifteen words. Then we treat the last clause of the joke as punchline and the rest as set-up. For de-duplication, we use the BOW (bag of words) and cosine similarity to detect the sentence similarity. Jokes with similarity greater than 0.93 are deleted.

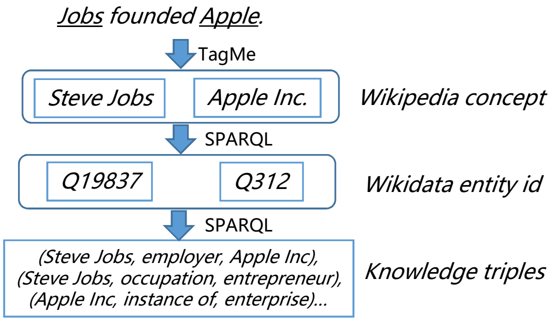

Figure 2: An example of knowledge acquisition for the sentence "Jobs founded Apple.". Firstly, we use TagMe for entity linking. After getting the Wikipedia concepts, We obtain knowledge triples through Wikidata Query Service.

To obtain background knowledge of the set-up sentences, we use the entity link tool TagMe (Ferragina and Scaiella, 2010). As an example shown in Fig. 4, TagMe can map entities in sentences to concepts in Wikipedia and give confidence of the mapping. To ensure the credibility of entities, we only keep entities with confidence greater than 0.1. After getting the entity's concepts on Wikipedia, we use SPARQL to link entities to Wikidata and get the entity-related triples. Overall, our dataset contains about 107,000 data pairs. We divide the training set, verification set and test set according to the 7:2:1 ratio.

4 Methodology

4.1 Problem Definition and Model Overview

We formulate the task of punchline generation with the set-up and relative knowledge. One knowledge triple is composed of subject s, relation r and object o, denoted as $k = (s, r, o)$. Given a set-up sentence $\mathbf{X} = \{x_1, x_2, \dots, x_p\}$ and its background knowledge triples $\mathbf{K} = \{k_1, k_2, \dots, k_u\}$, our goal is to generate a punchline $\mathbf{Y} = \{y_1, y_2, \dots, y_q\}$.

Our model is based on Transformer (Vaswani et al., 2017). The overview is shown in Fig. 3. Compared with the origin Transformer structure, we add two modules, knowledge encoder and knowledge fusion layer. Knowledge encoder obtains the hidden features of background knowledge. The knowl-

[1] https://www.kaggle.com/abhinavmoudgil95/short-jokes

[2] https://www.kaggle.com/cuddlefish/reddit-rjokes

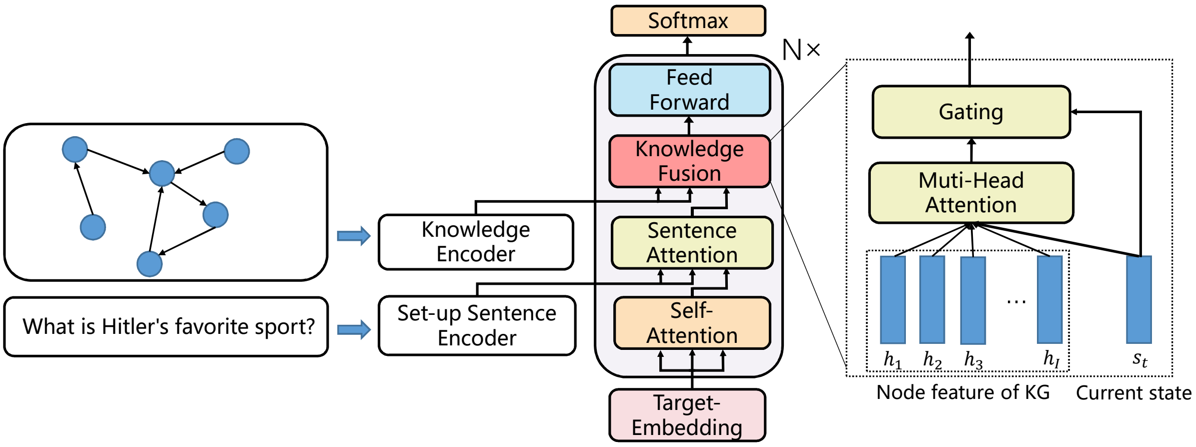

Figure 3: The overview of the proposed framework, which consists of a knowledge encoder, a set-up sentence encoder and a decoder with knowledge fusion layer.

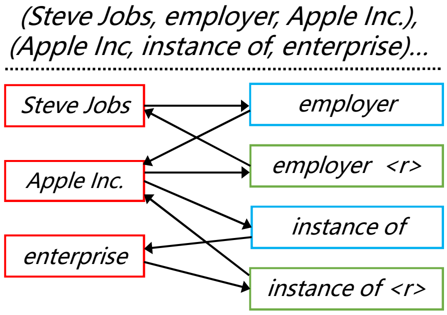

Figure 4: An example of constructing knowledge graph. Top: knowledge triples. Bottom: knowledge graph. The red, blue, and green boxes represent entity, forward relation, and reverse relation nodes, respectively. The reverse relation nodes are identified by the symbol $< r >$.

edge fusion layer fuses knowledge features into the decoding process after the multi-head attention layer in the decoder.

4.2 Constructing Knowledge Graph

Given a knowledge triple set $\mathbf{K} = \{k_1, k_2, \ldots, k_u\}$, we turn it into a directed graph. An example is shown in Fig. 4. Specifically, the co-referential entities in set $\mathbf{K}$ are folded into a single entity node, and the relations are mapped into relation nodes (The entity here is the subject and object). The subject, relation, and object nodes in one triple are connected in turn. In order to allow the information of the object to flow into the subject node, we add a reverse relation node which is similar to Koncel-Kedziorski et al. (2019). Since entities and relationships in Wikidata are usually multi-word ‘expressions, we encode these words with Bi-directional Long-Short Term Memory (Bi-LSTM) (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997). We adopt the last hidden states as the initial features of nodes. Finally, we get a connected graph $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{H}^0)$, where $\mathbf{V}$ is the set of nodes, $\mathbf{E}$ is the set of edges, $\mathbf{H}^0$ is the initial feature set of $\mathbf{V}$.

4.3 Knowledge Encoder

We use graph attention network (Velickovic et al., 2018) to incorporate the features of adjacent nodes in $\mathbf{G}$. For a knowledge graph $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{H}^l), \mathbf{V} = \{v_1, v_2, \ldots, v_I\}, \mathbf{H}^l = \{h_1^l, h_2^l, \ldots, h_I^l\}$, the initialization feature of node v_i is h_i^0. Each node updates feature through M-headed self-attention by receiving information from its neighbors, which can be described as follows.

$$h_i^{(l+1)} = \|_{m=1}^{M} \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij}^m \mathbf{W}_V^m h_j^l\right), \tag{1}$$

$$\alpha_{ij}^m = \frac{\exp\left(\left(\mathbf{W}_K^m h_j^l\right)^\top \mathbf{W}_Q^m h_i^l\right)}{\sum_{j \in \mathcal{N}_{(i)}} \exp\left(\left(\mathbf{W}_K^m h_j^l\right)^\top \mathbf{W}_Q^m h_i^l\right)}, \tag{2}$$

where the feature of node i in layer l is h_i^l, $h_i^l \in \mathbb{R}^d$. M is the number of heads, $\|$ denotes the concatenation of M attention heads. $\mathcal{N}(i)$ is one-hop neighbors of v_i (include v_i), σ is an activation function. $\mathbf{W}_Q^m, \mathbf{W}_K^m, \mathbf{W}_V^m \in \mathbb{R}^{d \times (d/M)}$ map h_i^l and h_j^l to the m-th head subspace, and we calculate the connection score α_{ij}^m by Eq. (2).

4.4 Decoder with Knowledge Fusion Layer

Before the knowledge fusion layer, the decoder's operation is the same as the original Transformer. Assume that the feature of nodes obtained by the knowledge encoder is $\mathbf{H} = \{h_1, h_2, \ldots, h_I\}$, and the input sequence of the decoder at time t is $\mathbf{Y}_t = \{y_0, y_1, \ldots, y_t\}$. We use a stack of N identical blocks to compute target-side representations. Each block is composed of four sub-layers as shown in Fig. 3. In n-th block, after the masked multi-head attention calculation with set-up sentence, the hidden state is expressed as $\mathbf{S}^n = \{s_1^n, s_2^n, ...s_t^n\}$.

The knowledge fusion layer contains a multi-head attention layer (Vaswani et al., 2017) and a gating machine inspired by highway network (Srivastava et al., 2015). Firstly, we integrate knowledge feature into the current state.

$$\mathbf{A}^n = \mathrm{MultiHead}(\mathbf{S}^n, \mathbf{H}, \mathbf{H}). \qquad (3)$$

Note that the node information in the background knowledge graph may contain noise due to the inaccuracy of entity link tools. To address this problem, we introduce the gating mechanism to allow for a better trade-off between the impact of background knowledge and the information from set-up encoder.

$$\mathrm{Gate}(\mathbf{S}^n) = \lambda^n \mathbf{S}^n + (1 - \lambda^n)\mathbf{A}^n, \qquad (4)$$

where λ denotes the gating weight, which is given by

$$\lambda^n = \mathrm{Sigmoid}(\mathbf{W}_g^n \mathbf{S}^n), \qquad (5)$$

where $\mathbf{W}_g$ is a model parameter.

Then we input the feature to the feed-forward layer of the Transformer. After the operations of N blocks, we get the final state $\{e_1, e_2, \ldots, e_t\}$. Finally, the probability distribution of generating the next target word y_{t+1} can be expressed as:

$$P\left(y_{t+1} | \mathbf{X}, \mathbf{K}, y_{<=t}; \theta\right) \propto \exp\left(\mathbf{W}_o e_t\right), \qquad (6)$$

where $\mathbf{W}_0 \in \mathbb{R}^{|\mathcal{V}_y| \times d}$ is a model parameter, $|\mathcal{V}_y|$ is the target vocabulary size.

5 Experiments

5.1 Evaluation

We use ROUGE-1, ROUGE-2, and ROUGE-L as automatic evaluation metrics, which measure the similarity between the output and the reference. We also conduct a human evaluation. For each model, we randomly select 40 set-ups and relative knowledge from the test set to generate punchlines. For further comparison, we also involve 40 human-written jokes. We invite 5 evaluators who are good at English and have the proper world knowledge to rate these jokes. We set three standards for evaluators to judge the punchlines: (1) Grammar and fluency (Is the punchline written in well-formed English?); (2) Coherency (Is the punchline coherent with the set-up sentence?); (3) Funniness (Is the punchline funny?). The score of each aspect ranges from 1 to 5, with the higher score the better.

5.2 Baselines and Implementation Details

Since there is no direct related work of this task, we compared three widely used text generation methods, including S2S-GRU with attention mechanism (Bahdanau et al., 2015), CopyNet (Gu et al., 2016), and Transformer (Vaswani et al., 2017). By comparing these models, it is shown that our method can use knowledge to enhance punchline generation.

In pre-processing, we use the pre-trained BPE dictionary with the vocabulary size of 25000 from Heinzerling and Strube (2018). Knowledge encoder uses 2 layers. For Transformer baseline and our model, the embedding and the hidden dimensions are 512. The block number of encoder and decoder is set to 4, the number of attention heads is set to 8, and the size of feed-forward layers is set to 2048. For S2S-GRU and CopyNet, the embeddings of words are 256 dimensions. We use 1-layer bidirectional GRU (Cho et al., 2014) with the hidden size of 256 as encoder. The decoder is a 2-layer GRU with the hidden size of 256. We also tried to increase the number of parameters of the S2S-GRU and CopyNet, but we did not get better results.

For our model, two-step training strategy is employed inspired by Zhang et al. 2018. Specially, we firstly pre-train a standard Transformer, which is used to initialize the parameters of set-up encoder and partial decoder. Then we fine-tun the entire model. During training, we use the Adam optimization (Kingma and Ba, 2015) with 16 mini-batch size. The learning rate is set to 0.001. During decoding, we implement beam search with beam size 5.

5.3 Result

Tab. 2 shows the automatic evaluation results. We find that: (1) As expected, Transformer based meth-

Set-up	Trump wants to cut funding for birth control, renegotiate trade deals and stop the wars.
Knowledge	(Donald Trump, field of work, politics); (Donald Trump, position held, President of the United States); (Birth control, part of, human population planning) . . .
S2S-GRU	He is not in denial.
CopyNet	It was not a solution.
Transformer	They are making headlines.
Our model	It seems he is a really nice president.
Human-written	It seems pulling out is his solution for everything.
Set-up	Cocaine makes people happy, what's the most dangerous thing about it?
Knowledge	(Cocaine, instance of, drug); (Cocaine,medical condition treated, pain); (Cocaine, Description, strong stimulant used as a recreational drug) . . .
S2S-GRU	Be important.
CopyNet	Happy life with danger.
Transformer	It seems to be more safe.
Our model	It is a drug.
Human-written	Maybe getting caught by the police

Table 1: Example outputs with four different models. Our model can generate more coherency punchlines.

Method	ROUGE-1	ROUGE-2	ROUGE-L
S2S-GRU	22.79	5.35	19.85
CopyNet	22.31	4.66	20.29
Transformer	23.73	6.27	21.89
Our model	25.97	9.47	23.60

Table 2: Automatic evaluations of generation models.

Method	Fluency	Coherency	Funniness
S2S-GRU	2.84	2.02	2.16
CopyNet	2.60	2.48	2.04
Transformer	3.04	2.86	2.40
Our model	3.24	3.28	2.60
Human-written	4.06	3.88	3.30

Table 3: Human evaluation of generation models.

ods perform better than other baselines. (2) Background knowledge can promote punchline generation, by comparing our method with origin Transformer.

Human evaluation results are shown in Tab. 3. Our method performs better than three baselines in all metrics. These results demonstrate the effectiveness of our model with knowledge enhanced. Nevertheless, there is still a gap between generated punchlines and expert-written punchlines across all aspects, indicating that humor generation remains an open challenge. Interestingly, the funniness score of human-written jokes is not very high, due to different people's sensitivity to humor. It is consistent with Petrović and Matthews (2013).

5.4 Case Study

Tab. 1 shows examples of various model outputs for two particular test instances. In general, all models produce fluent punchlines, but the semantic coherency between generated punchlines and set-up sentences is poor. Compared with baselines, our proposed method can generate more fluent and coherent punchlines. The first example is related to political commentary, which is often satirical humor. To some extent, the outputs of our method contains the background information of "Trump" and "Cocaine" .

6 Conclusion and Future Work

In this paper, we make the first endeavor to generate punchline, a freer form of humor generation. Besides we propose a knowledge-enhance framework which is generic and novel. Experiments show our method can make use of knowledge to enhance punchline generation. Future work can improve the knowledge selection method and add the explicit features of humor to the model.

7 Acknowledgement

This work is supported by the National Key R&D Program of China under contract No. 2017YFB1002201, the National Natural Science Fund for Distinguished Young Scholar (Grant No. 61625204), and partially supported by the Key Program of National Science Foundation of China (Grant No. 61836006).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations*.

Kim Binsted, Benjamin Bergen, Seana Coulson, Anton Nijholt, Oliviero Stock, Carlo Strapparava, Graeme Ritchie, Ruli Manurung, Helen Pain, Annalu Waller, and Dave O'Mara. 2006. Computational humor. *IEEE Intelligent Systems*, 21(2):59–69.

William Bright. 1992. International encyclopedia. *Linguistics*.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Paolo Ferragina and Ugo Scaiella. 2010. TAGME: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

He He, Nanyun Peng, and Percy Liang. 2019. Pun generation with surprise. 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)*.

Benjamin Heinzerling and Michael Strube. 2018. Bpemb: Tokenization-free pre-trained subword embeddings in 275 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*.

Nabil Hossain, John Krumm, Lucy Vanderwende, Eric Horvitz, and Henry Kautz. 2017. Filling the blanks (hint: plural noun) for mad Libs humor. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text generation from knowledge graphs with graph transformers. 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)*.

Dayiheng Liu, Jie Fu, Pengfei Liu, and Jiancheng Lv. 2019. TIGS: An inference algorithm for text infilling with gradient search. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4146–4156, Florence, Italy. Association for Computational Linguistics.

Saša Petrović and David Matthews. 2013. Unsupervised joke generation from big data. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Victor Raskin. 2012. *Semantic mechanisms of humor*, volume 24. Springer Science & Business Media.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Advances in neural information processing systems*.

Alessandro Valitutti, Hannu Toivonen, Antoine Doucet, and Jukka M. Toivanen. 2013. "let everything turn well in your wife": Generation of adult humor using lexical constraints. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

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 *Advances in neural information processing systems*, pages 5998–6008.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *6th International Conference on Learning Representations*.

Zhiwei Yu, Jiwei Tan, and Xiaojun Wan. 2018. A neural approach to pun generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Jiacheng Zhang, Huanbo Luan, Maosong Sun, Feifei Zhai, Jingfang Xu, Min Zhang, and Yang Liu. 2018. Improving the transformer translation model with document-level context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.