NAIST-NICT-TIT WMT22 General MT Task Submission

Hiroyuki Deguchi[†,††] **Kenji Imamura**[††] **Masahiro Kaneko**[†††] **Yuto Nishida**[†]
Yusuke Sakai[†] **Justin Vasselli**[†] **Huy Hien Vu**[†] **Taro Watanabe**[†]
[†] Nara Institute of Science and Technology [†††] Tokyo Institute of Technology
[††] National Institute of Information and Communications Technology
[†]{deguchi.hiroyuki.db0, nishida.yuto.nu8, sakai.yusuke.sr9,
vasselli.justin_ray.vk4, vu.huy_hien.va9, taro}@is.naist.jp
[††]kenji.imamura@nict.go.jp, [†††]masahiro.kaneko@nlp.c.titech.ac.jp

Abstract

In this paper, we describe our NAIST-NICT-TIT submission to the WMT22 general machine translation task. We participated in this task for the English ↔ Japanese language pair. Our system is characterized as an ensemble of Transformer big models, k-nearest-neighbor machine translation (kNN-MT) (Khandelwal et al., 2021), and reranking.

In our translation system, we construct the datastore for kNN-MT from back-translated monolingual data and integrate kNN-MT into the ensemble model. We designed a reranking system to select a translation from the n-best translation candidates generated by the translation system. We also use a context-aware model to improve the document-level consistency of the translation.

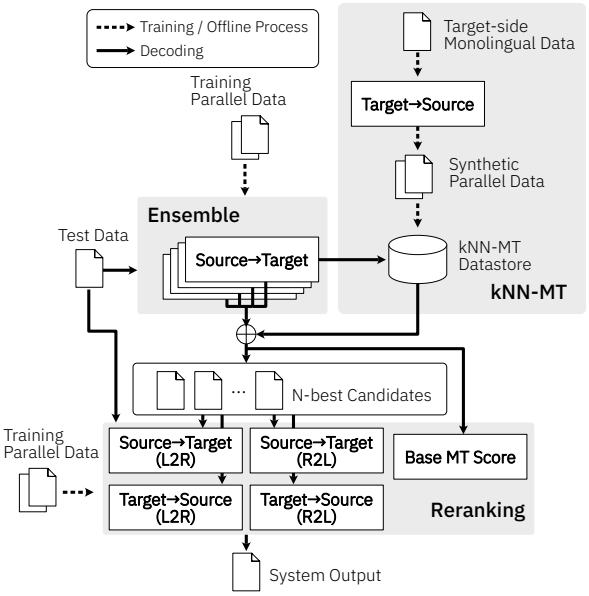

Figure 1: System overview.

1 Introduction

We participated in the WMT22 general machine translation task in two language directions, English-to-Japanese (En-Ja) and Japanese-to-English (Ja-En). We built our system on an ensemble of Transformer big models, k-nearest-neighbor machine translation (kNN-MT) (Khandelwal et al., 2021), and reranking. Figure 1 shows an overview of our system.

Our translation system is a combination of kNN-MT and an ensemble of four Transformer big models. We train each of the Transformer NMT models using a different random seed, and pick one model as kNN-MT. A notable point about our system is that we construct the datastore for kNN-MT from back-translated monolingual data rather than reusing training data. We found that using back-translated data improves translation performance compared with using a parallel training corpus for the datastore.

Our reranker is designed to select the translation from the n-best translation candidates generated by the translation system. The reranker computes the weighted sum of each translation candidate across multiple models and selects the translation candidate with the highest score. We used k-best batch MIRA (Cherry and Foster, 2012) to select the weights for the model scores that maximize the BLEU score of the development set.

2 Corpora and Preprocessing

For the training data, we used all the provided bilingual parallel data: JParaCrawl v3 (Morishita et al., 2020), News Commentary v16 (Tiedemann, 2012), Wiki Titles v3, WikiMatrix, Japanese-English Subtitle Corpus (Pryzant et al., 2018), The Kyoto Free Translation Task Corpus, and TED Talks. As the English translation of the Japanese-English Subtitle Corpus is only available in lowercase, we trained a Moses truecaser (Koehn et al., 2007) using the other corpora to add capitalization into the subtitle corpus. After truecasing, the first letter of each sentence was capitalized using detruecasing to produce sentence-cased English text

Proceedings of the Seventh Conference on Machine Translation (WMT), pages 244 - 250

to match the casing in the other corpora. We cleaned the data by removing duplicate lines and applying language filtering. As much of the training data was crawled from the internet, we used fasttext (Joulin et al., 2016a,b) to predict the language of each sentence and removed the sentences that were not predicted to be the correct language. This has the effect of reducing the noise of the dataset by removing sentences with garbage tokens. We tokenized the text into subword units using a joint vocabulary size of 64,000, a character coverage of 99.98%, and byte fallback using sentencepiece (Kudo and Richardson, 2018). After subword segmentation, all sentences shorter than 1 token or longer than 250 tokens were removed. We also removed all sentences where the number of tokens in one language was more than double the number of tokens in the translation, i.e the ratio of tokens between the source and target is >2.0. After filtering, 27,784,519 sentence pairs remained for training.

3 Translation System

3.1 Base Model

Our translation model is based on the Transformer big architecture (Vaswani et al., 2017) with an FFN size of 8,192 implemented in FAIRSEQ (Ott et al., 2019). The hyperparameters for our translation models are shown in Table 1.

Translation Model	
Architecture	Transformer big with FFN size of 8,192
Optimizer	Adam ($\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 1 \times 10^{-8}$)
Learning Rate Schedule	Inverse square root decay
Warmup Steps	4,000
Max Learning Rate	0.001
Dropout	0.3
Gradient Clipping	1.0
Label Smoothing	$\epsilon_{ls} = 0.1$
Mini-batch Size	512,000 tokens
Number of Updates	80,000 steps
Averaging	Save checkpoint for every 1,000 steps and take an average of last 10 checkpoints
Length Penalty	1.0
Beam Size	10
Reranker Model	
Architecture	Transformer big with FFN size of 8,192
Optimizer	Adam ($\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 1 \times 10^{-8}$)
Learning Rate Schedule	Inverse square root decay
Warmup Steps	4,000
Max Learning Rate	0.001
Dropout	0.3
Position Embeddings	SHAPE ($K = 512$)
Gradient Clipping	1.0
Label Smoothing	$\epsilon_{ls} = 0.1$
Mini-batch Size	512,000 tokens
Number of Updates	80,000 steps
Averaging	Save checkpoint for every 2,000 steps and take an average of last 10 checkpoints

Table 1: Hyperparameters of our translation and reranker models.

3.2 kNN-MT

kNN-MT (Khandelwal et al., 2021) extends the decoder of a trained machine translation model using the k-nearest-neighbor search algorithm, and retrieves the cached translation examples. The method consists of two steps, *datastore creation*, which creates key-value translation memory, and *generation*, which calculates an output probability distribution based on the nearest neighbors of cached translation memory.

Datastore creation The typical NMT model is composed of an encoder that encodes the source sentence X and a decoder that generates target tokens $Y = (y_1, y_2, \ldots, y_I)$. Each target token y_i is generated based on its output probability $P(y_i|X, y_{<i})$. kNN-MT caches parallel text $\mathcal{D}$ in a datastore represented as key-value memory $\mathcal{M}$. The value is a token y_i that comes from a target sentence in a parallel corpus, and the key is its intermediate vector h_i of each time step computed by the decoder. The datastore is formulated as follows:

$$\mathcal{M} = \{(h_i, y_i), \forall y_i \in Y \mid (X, Y) \in \mathcal{D}\}. \quad (1)$$

In our model, we use the 1024-dimensional vector representation from the decoder before it is passed to the final feed-forward network as the key h_i.

We use FAISS (Johnson et al., 2019), which is a toolkit for kNN search, to represent the datastore and search for the nearest neighbors. We use the `OPQMatrix` vector transform, `IndexIVFPQ` index, and `IndexFlatL2` index as the coarse quantizer. The hyperparameters of our search index are shown in Table 2.

Generation During decoding, kNN-MT generates output probabilities by computing the linear interpolation between the kNN and MT probabil-

Type	Value
Shape of OPQ matrix	$\mathbb{R}^{1024\times1024}$
Number of clusters (IVF)	65,536
Number of sub-vectors (PQ)	64
Number of clusters to search	64
Number of top-k neighbors	16

Table 2: Hyperparameters of our search index.

ity distributions,

$$P(y_i|X, y_{<i}, \theta) = \lambda p_{\text{kNN}}(y_i|X, y_{<i}, \theta) + (1-\lambda)p_{\text{MT}}(y_i|X, y_{<i}, \theta), \quad (2)$$

where λ is a hyperparameter for weighting each probability and θ represents the trained weight parameters and we set $\lambda = 0.4$

The k-nearest-neighbor keys $\mathcal{N}$ are converted into a distribution over the vocabulary p_{kNN} by applying softmax function.

$$p_{\text{kNN}}(y_i|X, y_{<i}, \theta) \propto \sum_{(k_j,v_j)\in\mathcal{N}} \mathbb{1}_{y_i=v_j} \exp\left(\frac{-||k_j - h_i||_2^2}{\tau}\right), \quad (3)$$

where k_j and v_j are the top-j neighbor key and value respectively and τ is a hyperparameter that represents the temperature of softmax and we set $\tau = 100$.

3.3 Back-Translated Monolingual Datastore

Our kNN-MT system uses back-translated monolingual data for the datastore instead of bilingual corpora. This method allows us to use monolingual resources without any additional training. First, we use the bilingual corpora to train a target-to-source translation model, which is then used to back-translate the monolingual corpora. The back-translated synthetic source sentences are then passed through the source-to-target model, which generates the intermediate vectors for each decoding time step to fill the datastore.

The monolingual data for English was taken from News Commentary v16 (Tiedemann, 2012), Europarl v10, Leipzig's news corpora (2018-2020), news-typical (2016), newscrawl and newscrawl-public (2018), web and web-public (2018-2020), and the largest available size of wikipedia corpus from each year for a total of over 26 million sentences. The monolingual data for Japanese includes News Commentary v16 (Tiedemann, 2012), and all Leipzig news, newscrawl, web, web-public, and wikipedia corpora from 2014 to 2021, totaling over 15 million sentences.

	Japanese	English
# of sentences	15,051,874	26,237,110
# of tokens	396,647,042	690,734,548

Table 3: Monolingual data statistics.

We preprocessed the monolingual data much the same way as the bilingual data, removing duplicate lines and using fasttext to filter out sentences where the predicted language did not match the target language. The text was tokenized into subword units using the model trained on the bilingual corpora. In order to reduce the time for the back-translation, sentences with more than 200 tokens were removed from the monolingual data. Table 3 shows the monolingual data statistics after preprocessing. Note that the number of tokens is equal to the size of the resulting kNN datastore for each target language.

3.4 kNN-MT with Ensemble Model

We integrate kNN-MT into the ensemble model. We train Transformer big models with different random seeds, just as we would build a normal ensemble model. Because the kNN search is too computationally expensive, we randomly pick a model instance and use it for the search as follows:

$$P(y_i|X, y_{<i}, \theta_1, \ldots, \theta_M) = \lambda p_{\text{kNN}}(y_i|X, y_{<i}, \theta_1) + \frac{1-\lambda}{M}\sum_{m=1}^{M} p_{\text{MT}}(y_i|X, y_{<i}, \theta_m), \quad (4)$$

where M is the number of model instances for the ensemble, and we set $M = 4$.

4 Reranker

Our reranker selects one of the n-best translation candidates from the translation system. Similar to other rerankers, it computes the weighted sum of multiple model scores for each translation candidate and selects the candidate with the maximum score. We used the average log-likelihoods of the

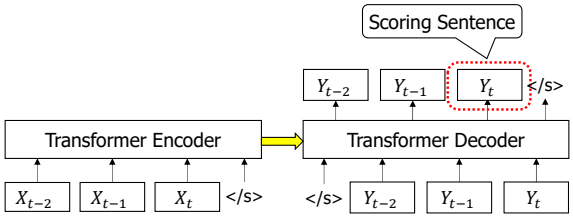

Figure 2: Context-aware model (context length $\ell = 2$).

words in each document as the model scores:

$$\hat{D} = \underset{Y_{t=1}^{T}}{\operatorname{argmax}} \left\{ \sum_k w_k \frac{\sum_{t=1}^{T} \sum_{i=1}^{|Y_t|} \mathcal{L}_k(y_{t,i})}{\sum_t |Y_t|} \right\}, \quad (5)$$

where D denotes a set of document translations, which consists of T translations ($D = Y_{t=1}^{T}$), and Y_t is the t-th translation in the document. $y_{t,i}$ denotes the i-th token in the translation Y_t, in which the number of tokens is $|Y_t|$. $\mathcal{L}_k(y_{t,i})$ denotes the log-likelihood of the token $y_{t,i}$ scored by the k-th model, and w_k is the weight of the k-th model.

The weights of the model scores were trained to maximize the BLEU score of the development set. We used k-best batch MIRA (Cherry and Foster, 2012) to optimize the weights.

4.1 Reranking Models

A characteristic of our reranker is the use of context-aware model scores, which are the log-likelihoods calculated per document of the test (or development) set. By taking the context into consideration during scoring, we expected to improve the consistency of the translation throughout each document.

We use an N-to-N translation model of Tiedemann and Scherrer (2017) as the context-aware model, which translates multiple concatenated sentences. Figure 2 illustrates the context-aware model in which the context length is two sentences ($\ell = 2$). The model computes the log-likelihood of the target translation Y_t using preceding ℓ sentences; that is,

$$\mathcal{L}_k(y_{t,i}) = \log p_k(y_{t,i} | X_{t-\ell}^{t}, Y_{t-\ell}^{t}), \quad (6)$$

where $p_k(\cdot)$ denotes the likelihood computed by the k-th model, and $X_{t-\ell}^{t}$ and $Y_{t-\ell}^{t}$ denote the source sentences and their translations from $t - \ell$ to t, respectively.

We used five models in total: the score from our translation system, and a combination of source-to-target and target-to-source translation and left-to-right (L2R) and right-to-left (R2L) decoding directions.

We trained the R2L models by reversing the order of the target tokens. Although the order is the same during scoring, we reversed the target tokens after concatenating multiple sentences. Therefore, the sentence order of the target side becomes (Y_t, Y_{t-1}, Y_{t-2}). Note that the scoring sentence is the last sentence (Y_{t-2}), and the R2L models score sentences later than the L2R models.

4.2 Training and Reranking

We used only the parallel corpora described in Section 2 and trained Transformer big models (Vaswani et al., 2017) with an FFN size of 8,192 for reranking. However, the trained models were sentence-wise models because we did not use document information in the training corpora. To apply the sentence-wise models to the N-to-N translation, we modified it using the following techniques.

- We did not use sentence separators (e.g., '[SEP]' between sentences) because the sentence-wise models did not include such separators.

  In the reranking task, we know the target tokens in advance, and we can easily identify the tokens for the target sentence, which we are scoring, without the separators.

- In the N-to-N translation, we had to score long translations because we simply concatenated multiple sentences during inference using a sentence-wise model which was not aware of the concatenated sentences. To learn appropriate models for long translations from sentences, we used the shifted absolute position embeddings (SHAPE) (Kiyono et al., 2021) to make a model invariant to absolute positions. The maximum shift was 512.

For the hyperparameters for the reranking models, we used the same setting as Kiyono et al. (2020) (Table 1).

To search for the best translations while considering context, we applied the beam search method to search for the translations that satisfied Eq. (5). We used a beam width of 10.

	En-Ja	Ja-En
Single Model	23.17	24.67
+ Ensemble	23.82	25.41
+ kNN-MT	24.43	25.16
+ kNN-MT with Ensemble	**24.72**	**25.84**

Table 4: Ablation study of our translation system on newstest 2020 (BLEU %).

Datastore	En-Ja	Ja-En
No Datastore (w/o kNN-MT)	23.17	24.67
Training Data	22.76	24.79
BT Monolingual Data	**24.43**	**25.16**

Table 5: Comparison of the kNN-MT datastore on newstest 2020 (BLEU %)

Direction	Reranking	newstest 2020	newstest 2021
En-Ja	No Reranking	24.7	26.8
	$\ell = 0$	24.8	27.3
	$\ell = 2$ (submission)	24.9	**27.4**
	$\ell = 4$	**25.0**	27.3
	Oracle	29.6	31.8
Ja-En	No Reranking	**25.6**	22.5
	$\ell = 0$	25.5	22.7
	$\ell = 2$ (submission)	25.5	**22.8**
	$\ell = 4$	25.5	22.7
	Oracle	29.8	25.7

Table 6: BLEU scores according to the reranking method.

5 Results

5.1 Translation System

Ablation Study To validate the effectiveness of our translation system, we performed an ablation experiment. Table 4 shows the experimental result on newstest 2020. Note that this experiment does not use a reranker system, and we evaluated the 1-best translation. The result shows that both ensemble and kNN-MT are effective, and combining them further improves translation performance.

kNN-MT Datastore As noted in Section 3.3, our kNN-MT datastore uses different data than the training corpus. We evaluated the translation performance of kNN-MT on a single Transformer model without ensemble. Table 5 shows the comparison of the kNN-MT datastore evaluated on newstest 2020. In the table, 'No Datastore' indicates that kNN-MT is not used, and 'Training Data' and 'BT Monolingaul Data' indicate that the datastore is constructed from training data and back-translated monolingual data, respectively. As shown in the table, our 'BT Monolingual Data' datastore outperforms the datastore constructed from the training data, despite its smaller size.

5.2 Context Length at Reranking

Table 6 shows the BLEU scores according to the reranking method. 'No Reranking' indicates the best translations output from the translation system. 'Oracle' always chooses the translation with the highest sentence BLEU score from the n-best translation candidates and represents the output of a perfect reranking system. The other cases indicate the BLEU scores of our reranker of varying context lengths ℓ.

The results show that our reranker improved the BLEU scores from the 'No Reranking' case, except for the case of Ja-En in newstest2020. However, the context length did not affect the BLEU scores. (We submitted the case of $\ell = 2$.) The BLEU scores of the reranker were still lower than that of 'Oracle', and future work will include studying the context-aware models to improve it further.

5.3 Placeholders

This year, the test set for the General MT task contained a set of placeholder tags, which should be output without translation. However, the provided parallel corpora for the task did not contain these special tokens. To solve this problem, we built a training set with placeholders using the existing parallel corpora.

We focused on the WikiTitles corpus, which is a subset of the parallel corpora provided for the General MT task. Most bitexts in WikiTitles are named entities because the corpus was extracted from Wikipedia titles. We substituted the parts that matched the WikiTitles entries with the placeholders.

In detail, we only extracted WikiTitles entries of five characters or more in Japanese and 10 characters or more in English from the parallel cor-

Direction	Placeholder (PLH)	Source	Translation Base	Translation PLH
En-Ja	#NAME#	3	2	3
	#NUMBER#	2	1	2
	#PRS_ORG#	55	52	55
	#URL#	4	3	4
	BLEU	-	39.2	38.5
Ja-En	#Organization1#	1	0	1
	#Person#	1	1	1
	#Product1#	116	79	116
	#Product2#	42	26	40
	#Product3#	7	4	7
	#Product4#	2	2	2
	#Product5#	2	2	2
	#Product6#	2	2	2
	#URL#	5	5	5
	#URL1#	1	0	1
	BLEU	-	22.7	22.7

Table 7: Results of placeholder translation.

pora using the longest match, and substituted the matched parts with the placeholders. (We used only one placeholder type: '#PLACEHOLDER#'.) Note that we excluded parallel sentences in Wiki-Titles from the parallel corpora in advance. As a result, we obtained additional 1.7 million parallel sentences that contained the placeholders and used them for training and fine-tuning.

During fine-tuning the translation system, we set an unused token in the vocabulary to the placeholder tag, and fine-tuned our translation models with a combination of data from the original training data (without placeholders) and the new data with the placeholders for two additional epochs from the averaged checkpoints.

Our placeholder corpus contained only a single placeholder tag instead of the rich variety of tags contained in the test set. We resolved this during the translation of the test set by first replacing all placeholder tags with our placeholder (#PLACEHOLDER#) before translation. After translation, we identified and replaced our #PLACEHOLDER# tag with the original tag from the source sentence. In the case of multiple placeholder tags in the same sentence, we preserved the original order when converting them back into the test placeholder tag set.

Table 7 shows the results of placeholder translation; that is, the number of placeholders and the BLEU score for the wmttest2022 test set. 'Base' and 'PLH' indicate translation using the model without/with fine-tuning on the placeholder corpus, respectively. The 'Base' model failed to translate some placeholders because it processed the placeholders as strings and translated them after subword segmentation. By contrast, the 'PLH' model translated the placeholders almost perfectly. However, the model fine-tuned on the placeholder corpus did not improve the BLEU scores, and we submitted the result of the 'Base' model.

6 Conclusions

In this paper, we described our submission of the joint team of NAIST, NICT, and TIT (NAIST-NICT-TIT) to the WMT22 general MT task. We participated in this task for the En ↔ Ja translation. Our system is built on an ensemble of Transformer big models, kNN-MT with using monolingual data, and k-best batch MIRA reranker. We would like to investigate each method and further improve translation performance.

References

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 427–436, Montréal, Canada. Association for Computational Linguistics.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016a. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016b. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2021. Nearest neighbor machine translation. In *International Conference on Learning Representations (ICLR)*.

Shun Kiyono, Takumi Ito, Ryuto Konno, Makoto Morishita, and Jun Suzuki. 2020. Tohoku-AIP-NTT at WMT 2020 news translation task. In *Proceedings of the Fifth Conference on Machine Translation*, pages 145–155, Online. Association for Computational Linguistics.

Shun Kiyono, Sosuke Kobayashi, Jun Suzuki, and Kentaro Inui. 2021. SHAPE: Shifted absolute position

embedding for transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3309–3321, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *CoRR*, abs/1808.06226.

Makoto Morishita, Jun Suzuki, and Masaaki Nagata. 2020. JParaCrawl: A large scale web-based English-Japanese parallel corpus. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 3603–3609, Marseille, France. European Language Resources Association.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

R. Pryzant, Y. Chung, D. Jurafsky, and D. Britz. 2018. JESC: Japanese-English Subtitle Corpus. *Language Resources and Evaluation Conference (LREC)*.

Jörg Tiedemann and Yves Scherrer. 2017. Neural machine translation with extended context. In *Proceedings of the Third Workshop on Discourse in Machine Translation*, pages 82–92, Copenhagen, Denmark. Association for Computational Linguistics.

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).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.