Vocabulary Learning via Optimal Transport for Neural Machine Translation

Jingjing Xu[1], **Hao Zhou**[1], **Chun Gan**[1,2†], **Zaixiang Zheng**[1,3†], **Lei Li**[1]
[1]ByteDance AI Lab
[2]Math Department, University of Wisconsin–Madison
[3]Nanjing University
{xujingjing.melody,zhouhao.nlp,lileilab}@bytedance.com
cgan5@wisc.edu
zhengzx@smail.nju.edu.cn

Abstract

The choice of token vocabulary affects the performance of machine translation. This paper aims to figure out what is a good vocabulary and whether one can find the optimal vocabulary without trial training. To answer these questions, we first provide an alternative understanding of the role of vocabulary from the perspective of information theory. Motivated by this, we formulate the quest of vocabularization – finding the best token dictionary with a proper size – as an optimal transport (OT) problem. We propose **VOLT**, a simple and efficient solution without trial training. Empirical results show that VOLT outperforms widely-used vocabularies in diverse scenarios, including WMT-14 English-German and TED multilingual translation. For example, VOLT achieves almost 70% vocabulary size reduction and 0.5 BLEU gain on English-German translation. Also, compared to BPE-search, VOLT reduces the search time from 384 GPU hours to 30 GPU hours on English-German translation. Codes are available at https://github.com/Jingjing-NLP/VOLT.

1 Introduction

Due to the discreteness of text, vocabulary construction (vocabularization for short) is a prerequisite for neural machine translation (NMT) and many other natural language processing (NLP) tasks using neural networks (Mikolov et al., 2013; Vaswani et al., 2017; Gehrmann et al., 2018; Zhang et al., 2018; Devlin et al., 2019). Currently, sub-word approaches like Byte-Pair Encoding (BPE) are widely used in the community (Ott et al., 2018; Ding et al., 2019; Liu et al., 2020), and achieve quite promising results in practice (Sennrich et al., 2016; Costa-jussà and Fonollosa, 2016; Lee et al., 2017; Kudo and Richardson, 2018; Al-Rfou et al., 2019; Wang et al., 2020). The key idea of these approaches is selecting the most frequent sub-words (or word pieces with higher probabilities) as the vocabulary tokens. In information theory, these frequency-based approaches are simple forms of data compression to reduce entropy (Gage, 1994), which makes the resulting corpus easy to learn and predict (Martin and England, 2011; Bentz and Alikaniotis, 2016).

However, the effects of vocabulary size are not sufficiently taken into account since current approaches only consider frequency (or entropy) as the main criteria. Many previous studies (Sennrich and Zhang, 2019; Ding et al., 2019; Provilkov et al., 2020; Salesky et al., 2020) show that vocabulary size also affects downstream performances, especially on low-resource tasks. Due to the lack of appropriate inductive bias about size, trial training (namely traversing all possible sizes) is usually required to search for the optimal size, which takes high computation costs. For convenience, most existing studies only adopt the widely-used settings in implementation. For example, 30K-40K is the most popular size setting in all 42 papers of Conference of Machine Translation (WMT) through 2017 and 2018 (Ding et al., 2019).

In this paper, we propose to explore automatic vocabularization by simultaneously considering entropy and vocabulary size without expensive trial training. Designing such a vocabularization approach is non-trivial for two main reasons. First, it is challenging to find an appropriate objective function to optimize them at the same time. Roughly speaking, the corpus entropy decreases with the increase of vocabulary size, which benefits model learning (Martin and England, 2011). On the other side, too many tokens cause token sparsity, which hurts model learning (Allison et al., 2006). Second, supposing that an appropriate measurement is given, it is still challenging to

[†]This work is done during the internship at ByteDance AI Lab.

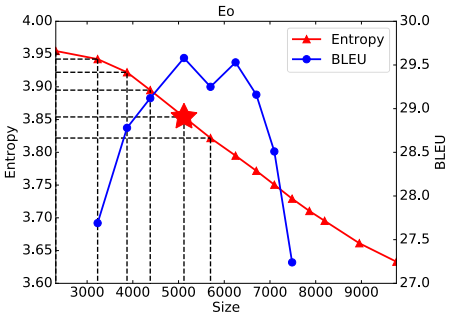

Figure 1: An illustration of marginal utility. We sample BPE-generated vocabularies with different sizes from Eo-En translation and draw their entropy (See Eq.2) and BLEU lines. "Star" represents the vocabulary with the maximum marginal utility. Marginal utility (See Eq.1) evaluates the increase of benefit (entropy decrease) from an increase of cost (size).

solve such a discrete optimization problem due to the exponential search space.

To address the above problems, we propose a **VO**cabulary **L**earning approach via optimal **T**ransport, VOLT for short. It can give an appropriate vocabulary in polynomial time by considering corpus entropy and vocabulary size. Specifically, given the above insight of contradiction between entropy and size, we first borrow the concept of *Marginal Utility* in economics (Samuelson, 1937) and propose to use *Marginal Utility of Vocabularization* (MUV) as the measurement. The insight is quite simple: in economics, marginal utility is used to balance the benefit and the cost and we use MUV to balance the entropy (benefit) and vocabulary size (cost). Higher MUV is expected for Pareto optimality. Formally, MUV is defined as the negative derivative of entropy to vocabulary size. Figure 1 gives an example about marginal utility. Preliminary results verify that MUV correlates with the downstream performances on two-thirds of tasks (See Figure 2).

Then our goal turns to maximize MUV in tractable time complexity. We reformulate our discrete optimization objective into an optimal transport problem (Cuturi, 2013) that can be solved in polynomial time by linear programming. Intuitively, the vocabularization process can be regarded as finding the *optimal transport matrix* from the *character distribution* to the *vocabulary token distribution*. Finally, our proposed VOLT will yield a vocabulary from the optimal transport matrix.

We evaluate our approach on multiple machine translation tasks, including WMT-14 English-German translation, TED bilingual translation, and TED multilingual translation. Empirical results show that VOLT beats widely-used vocabularies in diverse scenarios. Furthermore, VOLT is a lightweight solution and does not require expensive computation resources. On English-German translation, VOLT only takes 30 GPU hours to find vocabularies, while the traditional BPE-Search solution takes 384 GPU hours.

2 Related Work

Initially, most neural models were built upon word-level vocabularies (Costa-jussà and Fonollosa, 2016; Vaswani et al., 2017; Zhao et al., 2019). While achieving promising results, it is a common constraint that word-level vocabularies fail on handling rare words under limited vocabulary sizes.

Researchers recently have proposed several advanced vocabularization approaches, like byte-level approaches (Wang et al., 2020), character-level approaches (Costa-jussà and Fonollosa, 2016; Lee et al., 2017; Al-Rfou et al., 2019), and sub-word approaches (Sennrich et al., 2016; Kudo and Richardson, 2018). Byte-Pair Encoding (BPE) (Sennrich et al., 2016) is proposed to get subword-level vocabularies. The general idea is to merge pairs of frequent character sequences to create sub-word units. Sub-word vocabularies can be regarded as a trade-off between character-level vocabularies and word-level vocabularies. Compared to word-level vocabularies, it can decrease the sparsity of tokens and increase the shared features between similar words, which probably have similar semantic meanings, like "happy" and "happier". Compared to character-level vocabularies, it has shorter sentence lengths without rare words. Following BPE, some variants recently have been proposed, like BPE-dropout (Provilkov et al., 2020), SentencePiece (Kudo and Richardson, 2018), and so on.

Despite promising results, most existing sub-word approaches only consider frequency while the effects of vocabulary size is neglected. Thus, trial training is required to find the optimal size, which brings high computation costs. More recently, some studies notice this problem and propose some practical solutions (Kreutzer and Sokolov, 2018; Cherry et al., 2018; Chen et al., 2019; Salesky et al., 2020).

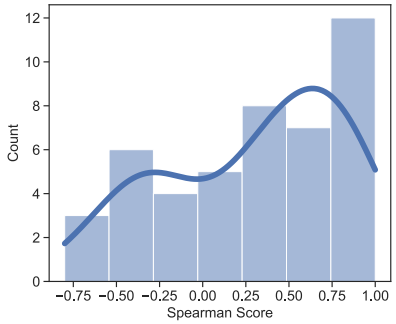

Figure 2: MUV and downstream performance are positively correlated on two-thirds of tasks. X-axis classifies Spearman scores into different groups. Y-axis shows the number of tasks in each group. The middle Spearman score is 0.4.

3 Marginal Utility of Vocabularization

In this section, we propose to find a good vocabulary measurement by considering entropy and size. As introduced in Section 1, it is non-trivial to find an appropriate objective function to optimize them simultaneously. On one side, with the increase of vocabulary size, the corpus entropy is decreased, which benefits model learning (Bentz and Alikaniotis, 2016). On the other side, a large vocabulary causes parameter explosion and token sparsity problems, which hurts model learning (Allison et al., 2006).

To address this problem, we borrow the concept of *Marginal Utility* in economics (Samuelson, 1937) and propose to use *Marginal Utility of Vocabularization* (MUV) as the optimization objective. MUV evaluates the benefits (entropy) a corpus can get from an increase of cost (size). Higher MUV is expected for higher benefit-cost ratio. Preliminary results verify that MUV correlates with downstream performances on two-thirds of translation tasks (See Figure 2). According to this feature, our goal turns to maximize MUV in tractable time complexity.

Definition of MUV Formally, MUV represents the negative derivation of entropy to size. For simplification, we leverage a smaller vocabulary to estimate MUV in implementation. Specially, MUV is calculated as:

$$\mathcal{M}_{v(k+m)} = \frac{-(\mathcal{H}_{v(k+m)} - \mathcal{H}_{v(k)})}{m}, \quad (1)$$

where $v(k)$, $v(k+m)$ are two vocabularies with k and $k+m$ tokens, respectively. $\mathcal{H}_v$ represents the corpus entropy with the vocabulary v, which is defined by the sum of token entropy. To avoid the effects of token length, here we normalize entropy with the average length of tokens and the final entropy is defined as:

$$\mathcal{H}_v = -\frac{1}{l_v}\sum_{i \in v} P(i)\log P(i), \quad (2)$$

where $P(i)$ is the relative frequency of token i from the training corpus and l_v is the average length of tokens in vocabulary v.

Preliminary Results To verify the effectiveness of MUV as the vocabulary measurement, we conduct experiments on 45 language pairs from TED and calculate the *Spearman correlation score*[*] between MUV and BLEU scores. We adopt the same and widely-used settings to avoid the effects of other attributes on BLEU scores, such as model hyper-parameters and training hyper-parameters. We generate a sequence of vocabularies with incremental sizes via BPE. All experiments use the same hyper-parameters. Two-thirds of pairs show positive correlations as shown in Figure 2. The middle Spearman score is 0.4. We believe that it is a good signal to show MUV matters. Please refer to Section 5 for more dataset details and Appendix A for more implementation details.

Given MUV, we have two natural choices to get the final vocabulary: search and learning. In the search-based direction, we can combine MUV with widely-used vocabularization solutions. For example, the optimal vocabularies can be obtained by enumerating all candidate vocabularies generated by BPE. While being simple and effective, it is not a self-sufficient approach. Furthermore, it still requires a lot of time to generate vocabularies and calculate MUV. To address these problems, we further explore a learning-based solution VOLT for more vocabulary possibilities. We empirically compare MUV-Search and VOLT in Section 5.

4 Maximizing MUV via Optimal Transport

This section describes the details of the proposed approach. We first show the general idea of VOLT in Section 4.1, then describe the optimal transport solution in Section 4.2, followed by the implementation details in Section 4.3.

[*] https://www.statstutor.ac.uk/resources/uploaded/spearmans.pdf

4.1 Overview

We formulate vocabulary construction as a discrete optimization problem whose target is to find the vocabulary with the highest MUV according to Eq. 1. However, the vocabulary is discrete and such discrete search space is too large to traverse, which makes the discrete optimization intractable.

In this paper, we simplify the original discrete optimization problem by searching for the optimal vocabulary from vocabularies with fixed sizes. Intuitively, MUV is the first derivative of entropy according to the vocabulary size (Eq. 1), and we introduce an auxiliary variable $\boldsymbol{S}$ ($\boldsymbol{S}$ is an *incremental* integer sequence) to approximate the computation by only computing MUV between vocabulary sizes as adjacent integers in $\boldsymbol{S}$.

Formally, $\boldsymbol{S} = \{i, 2\cdot i, ..., (t-1)\cdot i, \cdots\}$ where each timestep t represents a set of vocabularies with the number up to $\boldsymbol{S}[t]$. For any vocabulary, its MUV score can be calculated based on a vocabulary from its previous timestep. With sequence $\boldsymbol{S}$, the target to find the optimal vocabulary $v(t)$ with the highest MUV can be formulated as:

$$\begin{aligned}&\underset{v(t-1)\in\mathbb{V}_{\boldsymbol{S}[t-1]}, v(t)\in\mathbb{V}_{\boldsymbol{S}[t]}}{\arg\max} \mathcal{M}_{v(t)} = \\ &\underset{v(t-1)\in\mathbb{V}_{\boldsymbol{S}[t-1]}, v(t)\in\mathbb{V}_{\boldsymbol{S}[t]}}{\arg\max} -\frac{1}{i}\left[\mathcal{H}_{v(t)} - \mathcal{H}_{v(t-1)}\right],\end{aligned}$$

where $\mathbb{V}_{\boldsymbol{S}[t-1]}$ and $\mathbb{V}_{\boldsymbol{S}[t]}$ are two sets containing all vocabularies with upper bound of size $\boldsymbol{S}[t-1]$ and $\boldsymbol{S}[t]$. Due to exponential search space, we propose to optimize its lower bound:

$$\arg\max_{t} \frac{1}{i}\left[\max_{v(t)\in\mathbb{V}_{\boldsymbol{S}[t]}} \mathcal{H}_{v(t)} - \max_{v(t-1)\in\mathbb{V}_{\boldsymbol{S}[t-1]}} \mathcal{H}_{v(t-1)}\right]. \tag{3}$$

where i means the size difference between $t-1$ vocabulary and t vocabulary. MUV requires the size difference as a denominator. Based on this equation, the whole solution is split into two steps: 1) searching for the optimal vocabulary with the highest entropy at each timestep t; 2) enumerating all timesteps and outputing the vocabulary corresponding to the time step satisfying Eq. 3.

The first step of our approach is to search for the vocabulary with the highest entropy from $\mathbb{V}_{\boldsymbol{S}[t]}$. Formally, the goal is to find a vocabulary $v(t)$ such that entropy is maximized,

$$\arg\max_{v(t)\in\mathbb{V}_{\boldsymbol{S}[t]}} -\frac{1}{l_{v(t)}} \sum_{i\in v(t)} P(i)\log P(i), \tag{4}$$

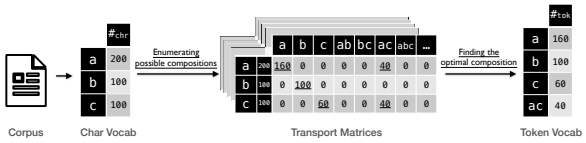

Figure 3: An illustration of vocabulary construction from a transport view. Each transport matrix represents a vocabulary. The transport matrix decides how many chars are transported to token candidates. The tokens with zero chars will not be added into the vocabulary.

where l_v is the average length for tokens in $v(t)$, $P(i)$ is the probability of token i. However, notice that this problem is in general intractable due to the extensive vocabulary size. Therefore, we instead propose a relaxation in the formulation of discrete optimal transport, which can then be solved efficiently via the Sinkhorn algorithm (Cuturi, 2013).

Intuitively, we can imagine vocabulary construction as a transport process that transports chars into token candidates with the number up to $\mathbf{S}[t]$. As shown in Figure 3, the number of chars is fixed, and not all token candidates can get enough chars. Each transport matrix can build a vocabulary by collecting tokens with chars. Different transport matrices bring different transport costs. The target of optimal transport is to find a transport matrix to minimize the transfer cost, i.e., negative entropy in our setting.

4.2 Vocabularization via Optimal Transport

Given a set of vocabularies $\mathbb{V}_{\boldsymbol{S}[t]}$, we want to find the vocabulary with the highest entropy. Consequently, the objective function in Eq. 4 becomes

$$\min_{v\in\mathbb{V}_{\boldsymbol{S}[t]}} \frac{1}{l_v}\sum_{i\in v} P(i)\log P(i),$$
$$\text{s.t.}\quad P(i) = \frac{\text{Token}(i)}{\sum_{i\in v}\text{Token}(i)},\ l_v = \frac{\sum_{i\in v} len(i)}{|v|}.$$

$\text{Token}(i)$ is the frequency of token i in the vocabulary v. $len(i)$ represents the length of token i. Notice that both the distribution $P(i)$ and the average length l_v depend on the choice of v.

Objective Approximation To obtain a tractable lower bound of entropy, it suffices to give a tractable upper bound of the above objective function. We adopt the merging rules to segment raw text similar with BPE where two consecutive tokens will be merged into one if the merged one is in the vocabulary. To this end, let $\mathbb{T} \in \mathbb{V}_{\boldsymbol{S}[t]}$ be

the vocabulary containing top $S[t]$ most frequent tokens, $\mathbb{C}$ be the set of chars and $|\mathbb{T}|, |\mathbb{C}|$ be their sizes respectively. Since $\mathbb{T}$ is an element of $\mathbb{V}_{\boldsymbol{S}[t]}$, clearly, we have

$$\min_{v \in \mathbb{V}_{\boldsymbol{S}[t]}} \frac{1}{l_v} \sum_{i \in v} P(i) \log P(i) \leq \frac{1}{l_{\mathbb{T}}} \sum_{i \in \mathbb{T}} P(i) \log P(i). \quad (5)$$

Here we start from the upper bound of the above objective function, that is $\frac{1}{l_{\mathbb{T}}} \sum_{i \in \mathbb{T}} P(i) \log P(i)$ and then search for a refined token set from $\mathbb{T}$. In this way, we reduce the search space into the subsets of $\mathbb{T}$. Let $P(i, j)$ be the joint probability distribution of the tokens and chars that we want to learn. Then we have

$$\begin{aligned} \sum_{i \in \mathbb{T}} P(i) \log P(i) &= \sum_{i \in \mathbb{T}} \sum_{j \in \mathbb{C}} P(i, j) \log P(i) \\ &= \underbrace{\sum_{i \in \mathbb{T}} \sum_{j \in \mathbb{C}} P(i, j) \log P(i, j)}_{\mathcal{L}_1} \\ &+ \underbrace{\sum_{i \in \mathbb{T}} \sum_{j \in \mathbb{C}} P(i, j)(-\log P(j|i))}_{\mathcal{L}_2}. \end{aligned} \quad (6)$$

The details of proof can be found at Appendix C. Since $\mathcal{L}_1$ is nothing but the negative entropy of the joint probability distribution $P(i, j)$, we shall denote it as $-H(P)$.

Let $\boldsymbol{D}$ be the $|\mathbb{C}| \times |\mathbb{T}|$ matrix whose (i, j)-th entry is given by $-\log P(j|i)$, and let $\boldsymbol{P}$ be the joint probability matrix, then we can write

$$\mathcal{L}_2 = \langle \boldsymbol{P}, \boldsymbol{D} \rangle = \sum_i \sum_j \boldsymbol{P}(i, j) \boldsymbol{D}(i, j). \quad (7)$$

In this way, Eq. 6 can be reformulated as the following objective function which has the same form as the objective function in optimal transport:

$$\min_{\boldsymbol{P} \in \mathbb{R}^{m \times n}} \langle \boldsymbol{P}, \boldsymbol{D} \rangle - \gamma H(\boldsymbol{P}). \quad (8)$$

Setup of OT From the perspective of optimal transport, $\boldsymbol{P}$ can be regarded as the transport matrix, and $\boldsymbol{D}$ can be regarded as the distance matrix. Intuitively, optimal transport is about finding the best transporting mass from the char distribution to the target token distribution with the minimum work defined by $\langle \boldsymbol{P}, \boldsymbol{D} \rangle$.

To verify the validness of transport solutions, we add the following constraints. First, to avoid invalid transport between char j and token i, we set the distance to $+\infty$ if the target token i does not contain the char j. Otherwise, we use $\frac{1}{len(i)}$ to estimate $P(j|i)$ where $len(i)$ is the length of token i. Formally, the distance matrix is defined as

$$\boldsymbol{D}(i, j) = \begin{cases} -\log P(j|i) = +\infty, & \text{if } j \notin i \\ -\log P(j|i) = -\log \frac{1}{len(i)}, & \text{otherwise} \end{cases}$$

Furthermore, the number of chars is fixed and we set the sum of each row in the transport matrix to the probability of char j. The upper bound of the char requirements for each token is fixed and we set the sum of each column in the transport matrix to the probablity of token j. Formally, the constraints are defined as:

$$|\sum_j \boldsymbol{P}(i, j) - P(i)| \leq \epsilon, \quad (9)$$

and

$$\sum_i \boldsymbol{P}(i, j) = P(j). \quad (10)$$

Given transport matrix $\boldsymbol{P}$ and distance matrix $\boldsymbol{D}$, the final objective can be formulated as:

$$\begin{aligned} &\arg\min_{\boldsymbol{P} \in \mathbb{R}^{|\mathbb{C}| \times |\mathbb{T}|}} -H(\boldsymbol{P}) + \langle \boldsymbol{P}, \boldsymbol{D} \rangle, \\ &\text{s.t.} \quad \sum_i \boldsymbol{P}(i, j) = P(j), \quad |\sum_j \boldsymbol{P}(i, j) - P(i)| \leq \epsilon, \end{aligned}$$

with small $\epsilon > 0$. Figure 4 shows the details of optimal transport solution. Strictly speaking, this is an unbalanced entropy regularized optimal transport problem. Nonetheless, we can still use the generalized Sinkhorn algorithm to efficiently find the target vocabulary as detailed in Section 4.6 of Peyré and Cuturi (2019). The algorithm details are shown in Algorithm 1. At each timestep t, we can generate a new vocabulary associated with entropy scores based on the transport matrix $\boldsymbol{P}$. Finally, we collect these vocabularies associated with entropy scores, and output the vocabulary satisfying Eq. 3.

4.3 Implementation

Algorithm 1 lists the process of VOLT. First, we rank all token candidates according to their frequencies. For simplification, we adopt BPE-generated tokens (e.g. BPE-100K) as the token candidates. It is important to note that any segmentation algorithms can be used to initialize token candidates. Experiments show that different initialization approaches result in similar results. We simply adopt BPE-100K for bilingual translation and BPE-300K for multilingual translation in this work. All token candidates with their probabilities are then used to initialize $\mathbb{L}$ in Algorithm 1.

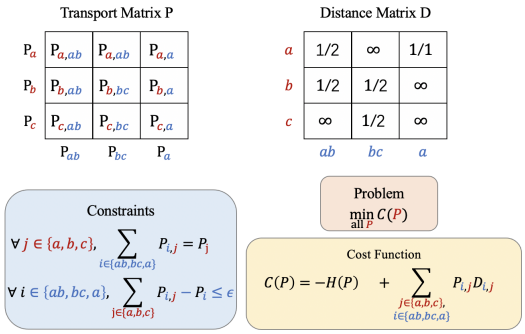

Figure 4: The details of optimal transport. The objective function is the sum of negative entropy and transport cost. Each element $D(i, j)$ in the distance matrix is the negative log of $1/n$ where n is the length of token i. It defines the distance between char j and token i. To avoid invalid transport between char j and token i, we set the distance to infinite if the target token i does not contain the char j.

The size of the incremental integer sequence $\boldsymbol{S}$ is a hyper-parameter and set to $(1K, ..., 10K)$ for bilingual translation, $(40K, ..., 160K)$ for multilingual settings. At each timestep, we can get the vocabulary with the maximum entropy based on the transport matrix. It is inevitable to handle illegal transport case due to relaxed constraints. We remove tokens with distributed chars less than 0.001 token frequencies. Finally, we enumerate all timesteps and select the vocabulary satisfying Eq. 3 as the final vocabulary.

After generating the vocabulary, VOLT uses a greedy strategy to encode text similar to BPE. To encode text, it first splits sentences into character-level tokens. Then, we merge two consecutive tokens into one token if the merged one is in the vocabulary. This process keeps running until no tokens can be merged. Out-of-vocabulary tokens will be split into smaller tokens.

5 Experiments

To evaluate the performance of VOLT, we conduct experiments on three datasets, including WMT-14 English-German translation, TED bilingual translation, and TED multilingual translation.

5.1 Settings

We run experiments on the following machine translation datasets. See Appendix B for more model and training details.

1. WMT-14 English-German (En-De) dataset: This dataset has 4.5M sentence pairs. The dataset is processed following Ott et al. (2018). We choose newstest14 as the test set.

2. TED bilingual dataset: We include two settings: X-to-English translation and English-to-X translation. We choose 12 language-pairs with the most training data. We use the language code according to ISO-639-1 standard[†]. TED data is provided by Qi et al. (2018).

3. TED multilingual dataset: We conduct experiments with 52 language pairs on a many-to-English setting. The network is trained on all language pairs. We adopt the same preprocessing pipeline in the WMT-14 En-De dataset.

Algorithm 1: VOLT

Input: A sequence of token candidates $\mathbb{L}$ ranked by frequencies, an incremental integer sequence $\boldsymbol{S}$ where the last item of $\boldsymbol{S}$ is less than $|\mathbb{L}|$, a character sequence $\mathbb{C}$, a training corpus D_c

Parameters: u $\in \mathbb{R}_+^{|\mathbb{C}|}$, v $\in \mathbb{R}_+^{|\mathbb{T}|}$

vocabularies = []

for *item in* $\boldsymbol{S}$ **do**

```
// Begin of Sinkhorn algorithm
```

Initialize u = ones() and v = ones()

$\mathbb{T} = \mathbb{L}[: item]$

Calculate token frequencies $P(\mathbb{T})$ based on D_c

Calculate char frequencies $P(\mathbb{C})$ based on D_c

Calculate $\boldsymbol{D}$

while *not converge* **do**

u = $P(\mathbb{T})/\boldsymbol{D}$v

v = $P(\mathbb{C})/\boldsymbol{D}^T$u

optimal_matrix = u.reshape(-1, 1) * $\boldsymbol{D}$ * v.reshape(1, -1)

```
// End of Sinkhorn algorithm
```

entropy, vocab = get_vocab(optimal_matrix)

vocabularies.append(entropy,vocab)

Output $\boldsymbol{v}^*$ from vocabularies satisfying Eq. 3

5.2 Main Results

Vocabularies Searched by VOLT are Better than Widely-used Vocabularies on Bilingual MT Settings. Ding et al. (2019) gather 42 papers that have been accepted by the research track of Conference of Machine Translation (WMT) through 2017 and 2018. Among these papers, the authors find that 30K-40K is the most popular range for the number of BPE merge actions. Following this work, we first compare our methods with dominant BPE-30K. The results are listed in Table 1. As we can see, the vocabularies searched by VOLT achieve higher BLEU scores with large

[†]http://www.lingoes.net/en/translator/langcode.htm

Table 1: Comparison between vocabularies search by VOLT and widely-used BPE vocabularies. VOLT achieves higher BLEU scores with large size reduction. Here the vocabulary size is adopted from the X-En setting.

Bilingual	WMT-14	TED											
En-X	De	Es	PTbr	Fr	Ru	He	Ar	It	Nl	Ro	Tr	De	Vi
BPE-30K	29.31	39.57	39.95	40.11	19.79	26.52	16.27	34.61	32.48	27.65	15.15	29.37	28.20
VOLT	**29.80**	**39.97**	**40.47**	**40.42**	**20.36**	**27.98**	**16.96**	**34.64**	**32.59**	**28.08**	**16.17**	**29.98**	**28.52**
X-En	De	Es	PTbr	Fr	Ru	He	Ar	It	Nl	Ro	Tr	De	Vi
BPE-30K	**32.60**	**42.59**	45.12	**40.72**	24.95	37.49	31.45	38.79	37.01	35.60	25.70	36.36	27.48
VOLT	32.30	42.34	**45.93**	**40.72**	**25.33**	**38.70**	**32.97**	**39.09**	**37.31**	**36.53**	**26.75**	**36.68**	**27.39**
Vocab Size (K)	De	Es	PTbr	Fr	Ru	He	Ar	It	Nl	Ro	Tr	De	Vi
BPE-30K	33.6	29.9	29.8	29.8	30.1	30.0	30.3	33.5	29.8	29.8	29.9	30.0	29.9
VOLT	**11.6**	**5.3**	**5.2**	**9.2**	**3.3**	**7.3**	**9.4**	**3.2**	**2.4**	**3.2**	**7.2**	**8.2**	**8.4**

Table 2: Comparison between vocabularies search by VOLT and BPE-1K, recommended by Ding et al. (2019) for low-resource datasets. Here we take TED X-En bilingual translation as an example. This table demonstrates that vocabularies searched by VOLT are on par with heuristically-searched vocabularies in terms of BLEU scores.

X-En	Es	PTbr	Fr	Ru	He	Ar	It	Nl	Ro	Tr	De	Vi	Avg
BPE-1K	**42.36**	45.58	**40.90**	24.94	38.62	32.23	38.75	**37.44**	35.74	25.94	**37.00**	27.28	35.65
VOLT	42.34	**45.93**	40.72	**25.33**	**38.70**	**32.97**	**39.09**	37.31	**36.53**	**26.75**	36.68	**27.39**	**35.81**
Vocab Size (K)	Es	PTbr	Fr	Ru	He	Ar	Ko	It	Nl	Ro	Tr	De	Avg
BPE-1K	**1.4**	**1.3**	**1.3**	**1.4**	**1.3**	**1.5**	**4.7**	**1.2**	**1.2**	**1.2**	**1.2**	**1.2**	**1.6**
VOLT	5.3	5.2	9.2	3.3	7.3	9.4	3.2	2.4	3.2	7.2	8.2	8.4	6.0

size reduction. The promising results demonstrate that VOLT is a practical approach that can find a well-performing vocabulary with higher BLEU and smaller size.

Vocabularies Searched by VOLT are on Par with Heuristically-searched Vocabularies on Low-resource Datasets. Ding et al. (2019) study how the size of BPE affects the model performance in low-resource settings. They conduct experiments on four language pairs and find that smaller vocabularies are more suitable for low-resource datasets. For Transformer architectures, the optimal vocabulary size is less than 4K, around up to 2K merge actions. We compare VOLT and BPE-1K on an X-to-English bilingual setting. The results are shown in Table 2. We can see that VOLT can find a good vocabulary on par with heuristically searched vocabularies in terms of BLEU scores. Note that BPE-1K is selected based on plenty of experiments. In contrast, VOLT only requires one trials for evaluation and only takes 0.5 CPU hours plus 30 GPU hours to find the optimal vocabulary.

VOLT Works Well on Multilingual MT Settings. We conduct a multilingual experiment. These languages come from multiple language families and have diverse characters. We compare VOLT with BPE-60K, the most popular setting in multilingual translation tasks. Table 3 lists the full results. The size of the searched vocabulary is around 110K. As we can see, VOLT achieves better BLEU scores on most pairs.

VOLT is a Green Vocabularization Solution. One advantage of VOLT lies in its low resource consumption. We compare VOLT with BPE-Search, a method to select the best one from a BPE-generated vocabulary set based on their BLEU scores. The results are shown in Table 4. In BPE-Search, we first define a vocabulary set including BPE-1K, BPE-2K, BPE-3K, BPE-4K, BPE-5K, BPE-6K, BPE-7K, BPE-8K, BPE-9K, BPE-10K, BPE-20K, BPE-30K. Then, we run full experiments to select the best vocabulary. Table 4 demonstrates that VOLT is a lightweight solution that can find a competitive vocabulary within 0.5 hours on a single CPU, compared to BPE-Search that takes hundreds of GPU hours. The cost of BPE-Search is the sum of the training time on all vocabularies. Furthermore, we also compare VOLT with MUV-Search as introduced in Section 3. MUV-Search is a method that combines MUV and popular approaches by selecting the vocabulary with the highest MUV as the final vocabulary.

Table 3: Comparison between VOLT and widely-used BPE vocabularies on multilingual translation. VOLT achieves higher BLEU scores on most pairs.

X-En	Es	Pt-br	Fr	Ru	He	Ar	Ko	Zh-cn	It	Ja	Zh-tw	Nl	Ro
BPE-60K	32.77	35.97	31.45	19.39	26.65	**22.28**	14.13	15.80	29.06	10.31	15.03	26.83	26.44
VOLT	**33.84**	**37.18**	**32.85**	**20.23**	**26.85**	22.17	**14.36**	**16.59**	**30.44**	**10.75**	**15.73**	**27.68**	**27.45**
X-En	Tr	De	Vi	Pl	Pt	Bg	El	Fa	Sr	Hu	Hr	Uk	Cs
BPE-60K	16.74	25.92	21.00	18.06	34.17	30.41	29.35	20.49	26.66	17.97	28.30	22.18	22.08
VOLT	**17.55**	**27.01**	**22.25**	**18.93**	**35.64**	**31.77**	**31.27**	**20.05**	**27.45**	**19.00**	**29.25**	**23.34**	**23.54**
X-En	Id	Th	Sv	Sk	Sq	Lt	Da	My	Sl	Mk	Fr-ca	Fi	Hy
BPE-60K	24.58	17.92	30.43	24.68	28.50	19.17	34.65	13.54	20.59	28.23	27.20	15.13	17.68
VOLT	**25.87**	**18.89**	**31.47**	**25.69**	**29.09**	**19.85**	**36.04**	**13.65**	**21.36**	**28.54**	**28.35**	**15.98**	**18.44**
X-En	Hi	Nb	Ka	Mn	Et	Ku	Gl	Mr	Zh	Ur	Eo	Ms	Az
BPE-60K	**18.57**	**35.96**	**16.47**	**7.96**	15.91	**13.39**	26.75	**8.94**	**13.35**	**14.21**	**21.66**	**19.82**	**9.67**
VOLT	18.54	35.88	15.97	**7.96**	**16.03**	13.20	**26.94**	8.40	12.67	13.89	21.43	19.06	9.09

Table 4: Results of VOLT, MUV-Search and BPE-Search. MUV-Search does not require full training and saves a lot of costs. Among them, VOLT is the most efficient solution. MUV-Search and VOLT require additional costs for downstream evaluation, which takes around 32 GPU hours. "GH" and "CH" represent GPU hours and CPU hours, respectively.

En-De	BLEU	Size	Cost
BPE-Search	**29.9**	12.6K	384 GH
MUV-Search	29.7	**9.70K**	5.4 CH + 30 GH
VOLT	29.8	11.6K	**0.5 CH + 30 GH**

Table 5: Comparison between VOLT and strong baselines. VOLT achieves almost the best performance with a much smaller vocabulary.

En-De	BLEU	Parameters
(Vaswani et al., 2017)	28.4	210M
(Shaw et al., 2018)	29.2	213M
(Ott et al., 2018)	29.3	210M
(So et al., 2019)	29.8	218M
(Liu et al., 2020)	**30.1**	256M
SentencePiece	28.7	210M
WordPiece	29.0	210M
VOLT	29.8	**188M**

Table 6: Vocabularies searched by VOLT are better than widely-used vocabularies on various architectures. Here "better" means competitive results but much smaller sizes.

En-De	Approach	BLEU	Size
Transformer-big	BPE-30K	29.3	33.6K
	VOLT	**29.8**	**11.6K**
Convolutional Seq2Seq	BPE-30K	**26.4**	33.6K
	VOLT	26.3	**11.6K**

We generate a sequence of BPE vocabularies with incremental size 1K, 2K, 3K, 4K, 5K, 6K, 7K, 8K, 9K, 10K, 20K. For t-th vocabulary $v(t)$, its MUV score is calculated according to $v(t)$ and $v(t-1)$. We enumerate all vocabularies and select the vocabulary with the highest MUV as the final vocabulary. The comparison between VOLT and MUV-Search is shown in Table 4. Although MUV-Search does not require downstream full-training, it still takes a lot of time to generate vocabularies and calculate MUV. Among them, VOLT is the most efficient approach.

5.3 Discussion

We conduct more experiments to answer the following questions: 1) can a baseline beat strong approaches with a better vocabulary; 2) can VOLT beat recent vocabulary solutions, like SentencePiece; 3) can VOLT work on diverse architectures?

A Simple Baseline with a VOLT-generated Vocabulary Reaches SOTA Results. We compare VOLT and several strong approaches on the En-De dataset. Table 5 shows surprisingly good results. Compared to the approaches in the top block, VOLT achieves almost the best performance with a much smaller vocabulary. These results demonstrate that a simple baseline can achieve good results with a well-defined vocabulary.

VOLT Beats SentencePiece and WordPiece. SentencePiece and WordPiece are two variants of sub-word vocabularies. We also compare our approach with them on WMT-14 En-De translation

to evaluate the effectiveness of VOLT. The middle block of Table 5 lists the results of SentenPiece and WordPiece. We implement these two approaches with the default settings. We can observe that VOLT outperforms SentencePiece and WordPiece by a large margin, with over 1 BLEU improvements.

VOLT Works on Various Architectures. This work mainly uses Transformer-big in experiments. We are curious about whether VOLT works on other architectures. We take WMT-14 En-De translation as an example and implement a Convolutional Seq2Seq model. The network uses the default settings from Fairseq[‡]. We set the maximum epochs to 100 and average the last five models as the final network for evaluation. Table 6 demonstrates that vocabularies searched by VOLT also works on Convolutional Seq2Seq with competitive BLEU but much smaller size. In this work, we verify the effectiveness of VOLT on architectures with standard sizes. Since model capacity is also an important factor on BLEU scores, we recommend larger vocabularies associated with more embedding parameters for small architectures.

VOLT can Bring Slight Speedup During Training. We evaluate the running time for VOLT vocabulary and BPE-30K on WMT En-De translation. The model with VOLT-searched vocabulary (11.6k tokens) can process 133 sentences per second, while the model with BPE-30K (33.6k tokens) only executes 101 sentences per second. All experiments run on the same environment (2 Tesla-V100-GPUs + 1 Gold-6130-CPU), with the same beam size for decoding. The speedup mainly comes from larger batch size with reduced embedding parameters. We also find that although VOLT reduces the Softmax computations, it does not significantly boost the Softmax running time due to optimized parallel computation in GPUs.

VOLT Vocabularies and BPE Vocabularies are Highly Overlapped. For simplification, VOLT starts from BPE-segmented tokens. We take WMT En-De as an example to see the difference between VOLT vocabulary and BPE vocabulary. The size of VOLT vocabulary is around 9K and we adopt BPE-9K vocabulary for comparison. We find that these two vocabularies are highly overlapped, especially for those high-frequency words. They also have similar downstream performance. Therefore, from an empirical perspective, BPE with VOLT size is also a good choice.

[‡]https://github.com/pytorch/fairseq/tree/master/examples/translation

6 Conclusion

In this work, we propose a new vocabulary search approach without trail training. The whole framework starts from an informtaion-therotic understanding. According to this understanding, we formulate vocabularization as a two-step discrete optimization objective and propose a principled optimal transport solution VOLT. Experiments show that VOLT can effectively find a well-performing vocabulary in diverse settings.

Acknowledgments

We thank the anonymous reviewers, Demi Guo, for their helpful feedback. Lei Li and Hao Zhou are corresponding authors.

References

Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. 2019. Character-level language modeling with deeper self-attention. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 3159–3166. AAAI Press.

Ben Allison, David Guthrie, and Louise Guthrie. 2006. Another look at the data sparsity problem. In *Text, Speech and Dialogue, 9th International Conference, TSD 2006, Brno, Czech Republic, September 11-15, 2006, Proceedings*, volume 4188 of *Lecture Notes in Computer Science*, pages 327–334. Springer.

Christian Bentz and Dimitrios Alikaniotis. 2016. The word entropy of natural languages. *arXiv preprint arXiv:1606.06996*.

Wenhu Chen, Yu Su, Yilin Shen, Zhiyu Chen, Xifeng Yan, and William Yang Wang. 2019. How large a vocabulary does text classification need? A variational approach to vocabulary selection. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 3487–3497. Association for Computational Linguistics.

Colin Cherry, George F. Foster, Ankur Bapna, Orhan Firat, and Wolfgang Macherey. 2018. Revisiting character-based neural machine translation with capacity and compression. In *Proceedings of the 2018*

Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018, pages 4295–4305. Association for Computational Linguistics.

Marta R. Costa-jussà and José A. R. Fonollosa. 2016. Character-based neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*. The Association for Computer Linguistics.

Marco Cuturi. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 2292–2300.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Shuoyang Ding, Adithya Renduchintala, and Kevin Duh. 2019. A call for prudent choice of subword merge operations in neural machine translation. In *Proceedings of Machine Translation Summit XVII Volume 1: Research Track, MTSummit 2019, Dublin, Ireland, August 19-23, 2019*, pages 204–213. European Association for Machine Translation.

Philip Gage. 1994. A new algorithm for data compression. *C Users Journal*, 12(2):23–38.

Sebastian Gehrmann, Yuntian Deng, and Alexander M. Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4098–4109. Association for Computational Linguistics.

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*.

Julia Kreutzer and Artem Sokolov. 2018. Learning to segment inputs for NMT favors character-level processing. *CoRR*, abs/1810.01480.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations, Brussels, Belgium, October 31 - November 4, 2018*, pages 66–71. Association for Computational Linguistics.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378.

Xiaodong Liu, Kevin Duh, Liyuan Liu, and Jianfeng Gao. 2020. Very deep transformers for neural machine translation. *CoRR*, abs/2008.07772.

Nathaniel FG Martin and James W England. 2011. *Mathematical theory of entropy*. 12. Cambridge university press.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers, WMT 2018, Belgium, Brussels, October 31 - November 1, 2018*, pages 1–9. Association for Computational Linguistics.

Gabriel Peyré and Marco Cuturi. 2019. Computational optimal transport. *Found. Trends Mach. Learn.*, 11(5-6):355–607.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2020. Bpe-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 1882–1892. Association for Computational Linguistics.

Ye Qi, Devendra Singh Sachan, Matthieu Felix, Sarguna Padmanabhan, and Graham Neubig. 2018. When and why are pre-trained word embeddings useful for neural machine translation? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pages 529–535. Association for Computational Linguistics.

Elizabeth Salesky, Andrew Runge, Alex Coda, Jan Niehues, and Graham Neubig. 2020. Optimizing segmentation granularity for neural machine translation. *Machine Translation*, pages 1–19.

Paul A Samuelson. 1937. A note on measurement of utility. *The review of economic studies*, 4(2):155–161.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Rico Sennrich and Biao Zhang. 2019. Revisiting low-resource neural machine translation: A case study. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 211–221. Association for Computational Linguistics.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pages 464–468. Association for Computational Linguistics.

David R. So, Quoc V. Le, and Chen Liang. 2019. The evolved transformer. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5877–5886. PMLR.

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 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

Changhan Wang, Kyunghyun Cho, and Jiatao Gu. 2020. Neural machine translation with byte-level subwords. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 9154–9160. AAAI Press.

Lei Zhang, Shuai Wang, and Bing Liu. 2018. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4).

Yi Zhao, Yanyan Shen, and Junjie Yao. 2019. Recurrent neural network for text classification with hierarchical multiscale dense connections. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 5450–5456. ijcai.org.

Appendix A: MUV

To evaluate the relationship between MUV and BLEU scores, we conduct experiments on 45 language pairs (X-En) with most resources (including ar-en, eg-en, cs-en, da-en, de-en, el-en, es-en, et-en, fa-en, fi-en, fr-ca-en, fr-en, gl-en, he-en, hi-en, hr-en, hu-en, hy-en, id-en, it-en, ja-en, ka-en, ko-en, ku-en, lt-en, mk-en, my-en, nb-en, nl-en, pl-en, pt-br-en, pt-en, ro-en, ru-en, sk-en, sl-en, sq-en, sr-en, sv-en, th-en, tr-en, uk-en, vi-en, zh-cn-en, zh-tw-en) from TED and calculate the Spearman correlation score beween MUV and BLEU. We merge all bilingual training data together and pre-train a multilingual network. To avoid the effects of unsteady BLEU scores, we use the multilingual network to initialize bilingual networks. All bilingual datasets are segment by four multilingual vocabularies, including BPE-20K, BPE-60K, BPE-100K, BPE-140K. In this way, we can get four bilingual corpora for each translation task. The MUV is calculated based on these corpora. For each corpus, we leverage a corpus with a smaller vocabulary to calculate MUV. For example, the MUV score of Ar-En (BPE-20K) is calculated based on Ar-En (BPE-20K) and Ar-En (BPE-10K). It is important to note that all corpora adopt the same interval, 10K, to calculate MUV. All bilingual datasets share the same model hyper-parameters and training hyper-parameters (Please refer to Appendix B for more implementation details). We set the maximum training epoch to 50 and average the last five models as the final network for evaluation.

Appendix B: Experiments

Models. We use Fairseq to train a Transformer-big model with the same setting in the original paper (Ott et al., 2018). The input embedding and output embeddings are shared. We use the Adam optimizer (Kingma and Ba, 2015) with a learning rate 5e-4 and an inverse_sqrt decay schedule. The warm-up step is $4,000$, the dropout rate is 0.3, the update frequency is 4, the number of tokens is $9,600$, or $4,800$ in a single batch.

Training and Evaluation. We run WMT-14 En-De experiments with 8 GPUs, TED bilingual translation with 4 GPUs, TED multilingual translation with 16 GPUs. We set a beamwidth to 4 for En-De and 5 for the other. For bilingual translation, we run approaches 40 epochs, average the last five models on all datasets, and use the averaged model to generate translation results. For multilingual translation, all approaches run 10 epochs and we adopt the last model for evaluation. We calculate case-sensitive tokenized BLEU for evaluation.

Appendix C: Proofs for Eq. 6

$$
\begin{aligned}
\sum_{i\in\mathbb{T}} P(i)\log P(i) &= \sum_{i\in\mathbb{T}}\sum_{j\in\mathbb{C}} P(i,j)\log P(i) \\
&= \sum_{i\in\mathbb{T}}\sum_{j\in\mathbb{C}} P(i,j)\log P(i,j)\cdot\frac{P(i)}{P(i,j)} \\
&= \sum_{i\in\mathbb{T}}\sum_{j\in\mathbb{C}} P(i,j)\log P(i,j) + \sum_{i\in\mathbb{T}}\sum_{j\in\mathbb{C}} P(i,j)\log\frac{P(i)}{P(i,j)} \\
&= \underbrace{\sum_{i\in\mathbb{T}}\sum_{j\in\mathbb{C}} P(i,j)\log P(i,j)}_{\mathcal{L}_1} + \underbrace{\sum_{i\in\mathbb{T}}\sum_{j\in\mathbb{C}} P(i,j)(-\log P(j|i))}_{\mathcal{L}_2}.
\end{aligned}
$$

□