Free Lunch for Efficient Textual Commonsense Integration in Language Models

Wanyun Cui
Shanghai University of Finance and Economics
cui.wanyun@shufe.edu.cn

Xingran Chen
University of Michigan
chenxran@umich.edu

Abstract

Recent years have witnessed the emergence of textual commonsense knowledge bases, aimed at providing more nuanced and context-rich knowledge. The integration of external commonsense into language models has been shown to be a key enabler in advancing the state-of-the-art for a wide range of NLP tasks. However, incorporating textual commonsense descriptions is computationally expensive, as compared to encoding conventional symbolic knowledge. In this paper, we propose a method to improve its efficiency without modifying the model. We group training samples with similar commonsense descriptions into a single batch, thus reusing the encoded description across multiple samples. One key observation is that the upper bound of batch partitioning can be reduced to the classic *graph k-cut problem*. Consequently, we propose a spectral clustering-based algorithm to solve this problem. Extensive experiments illustrate that the proposed batch partitioning approach effectively reduces the computational cost while preserving performance. The efficiency improvement is more pronounced on larger datasets and on devices with more memory capacity, attesting to its practical utility for large-scale applications.

1 Introduction

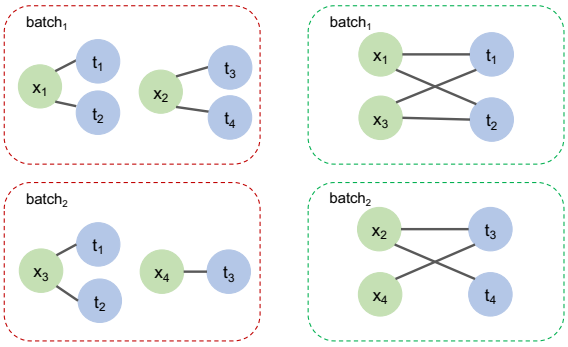
While pre-trained language models have made substantial progress in natural language processing, they still lack certain knowledge. Thus it is critical to incorporate external knowledge sources (Peters et al., 2019; Zhang et al., 2019; Logan et al., 2019). Previous research has primarily focused on incorporating symbolic knowledge from structured knowledge graphs. Recently, realizing the lack of expressiveness and contextualization of symbolic knowledge, many forms of commonsense knowledge bases are constructed, such as if-then knowledge (Sap et al., 2019) and discourse knowledge (Fang et al., 2021). The integration of such textual commonsense knowledge into language models has been shown to improve the state of the art for various tasks, such as named entity recognition (Wu et al., 2020) and commonsense knowledge base completion (Malaviya et al., 2020).

However, integrating such commonsense knowledge are computationally expensive. Commonsense knowledge in text form requires more complex encoders (e.g. Transformer (Vaswani et al., 2017)), as opposed to the simple lookup operation for discrete symbolic knowledge. The feed-forward and back-propagation process for the text encoder is significantly more computationally expensive than the standalone symbolic knowledge embeddings. Therefore, it is essential to reduce the computational cost for efficient integration of textual commonsense knowledge, particularly for large-scale applications.

In this paper, we propose a method to accelerate the process of incorporating textual commonsense knowledge into language models. Our approach is based on the observation that if multiple training samples in a mini-batch share the same commonsense description, the encoding for that description can be reused across those samples. In other words, we only need to encode each *distinct* description in a mini-batch once. For example, consider the training samples $x_{1\cdots4}$ and the associated commonsense $t_{1\cdots4}$ in Fig. 1. In the batch partitioning in Fig. 1a, the samples in one batch have no shared descriptions, requiring seven times of commonsense encoding for t_i. However, in the batch partitioning shown in Fig. 1b, each description will be encoded only once, resulting in only four times of encoding for t_i. The cost of encoding the commonsense is significantly reduced by effective partitioning of the training samples. Therefore, our goal is to group the training samples in such a way as to minimize the total number of distinct commonsense descriptions per mini-batch.

To optimize the batch partitioning, we begin by theoretically analyzing the objective (§2.1). Our

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 3759–3770



(a) If samples are divided randomly into batches, a total of **7** times of encoding for t_i is required.

(b) If samples are divided delicately, only a total of **4** times of encoding for t_i is required.

Figure 1: Idea of batch partitioning.

key observation is that the upper bound of the cost can be reduced to the well-studied *graph k-cut* problem (Rangapuram et al., 2014) (§ 3.1 § 3.2). As a result, we minimize the upper bound instead by adapting the classic spectral clustering algorithm (§ 3.3). The average distinct commonsense descriptions per batch are approximated by the distance to the cluster centroid, and is optimized by spectral clustering. This is also empirically verified (§ 5.4).

The main contributions of this paper are as follows: (1) We propose the use of batch partitioning for improving the efficiency of textual commonsense integration for language models. (2) We theoretically demonstrate that the batch partitioning problem can be reduced to the classic graph k-cut problem, and we use the well-studied spectral clustering to optimize it. (3) We empirically show that the efficiency of integrating commonsense descriptions can be significantly improved without sacrificing effectiveness. The acceleration is even more pronounced for large-scale training.

2 The Batch Partitioning Problem

In this section, we analyze the training efficiency w.r.t. batch partitioning. We first show in § 2.1 that the complexity of the model depends on the number of corresponding knowledge descriptions per sample. Then, in § 2.2, we formally define this batch partitioning problem.

2.1 Model Setup and Complexity Analysis

In this paper, we use the OK-Transformer (Cui and Chen, 2022) as the backbone. OK-Transformer is a recently proposed model that effectively introduces commonsense knowledge into language models. Traditional approaches for such introduction required pre-training language models on a large corpus along with external commonsense, which was time-consuming (Peters et al., 2019; Zhang et al., 2019). The OK-Transformer model, on the other hand, is able to directly incorporate extra knowledge without pre-training. This model utilizes commonsense tokens and attention mechanisms to effectively integrate textual commonsense. Our proposed batch partitioning method is also applicable to other models that encode target sentences and associated commonsense descriptions.

To analyze the computational complexity of encoding commonsense knowledge and formulate the problem, we briefly describe how the original OK-Transformer works. It consists of three Transformers, where $\text{Transformer}^{(1)}$ is used to represent the target sentence, $\text{Transformer}^{(2)}$ is used to represent each textual commonsense description, and $\text{Transformer}^{(3)}$ is used to incorporate commonsense embeddings from $\text{Transformer}^{(2)}$ into $\text{Transformer}^{(1)}$.

We now concretely analyze the complexity of integrating external textual commonsense. When encoding a sample with associated commonsense descriptions, the complexity consists of three modules:

- For encoding the target sentence via $\text{Transformer}^{(1)}$, the complexity of encoding a sentence of length L into dimension D is $O(L^2D)$.
- For encoding textual commonsense descriptions via $\text{Transformer}^{(2)}$, the complexity of encoding C knowledge descriptions of length L is $O(CL^2D)$.
- For integrating the knowledge embeddings into the target sentence via $\text{Transformer}^{(3)}$, the complexity is $O(C^2D)$.

Module	Complexity
Target sentence encoding	$O(L^2D)$
External knowledge encoding	$O(CL^2D)$
Knowledge integration	$O(C^2D)$

Table 1: Module complexities.

We summarize the complexity in Table 1. Since in practice we usually have $L^2 \gg C$, the key is is to reduce the complexity of encoding for textual commonsense descriptions, i.e., reduce $O(CL^2D)$.

Relation to retrieval-based knowledge incorporation Integrating text commonsense is related to learning dense retrievers for efficiently retrieving and introducing external textual commonsense, such as REALM (Guu et al., 2020). In commonsense incorporation, each sample only retrieves a small number of knowledge descriptions based on trigger words. So the key of our problem is to efficiently and effectively incorporate certain knowledge descriptions, rather than the information retrieval in dense retrievers. Specifically, dense retrievers typically consist of a retriever and a knowledge-augmented encoder. Our work can be analogous to reducing the cost of the knowledge-augmented encoder.

2.2 Problem Formulation

We now formulate the problem of batch partitioning. As stated in the introduction, different samples may correspond to the same textual commonsense description. We only need to encode the distinct commonsense descriptions once for a batch of samples. Therefore, the goal of batch partitioning is to minimize the number of distinct commonsense descriptions per batch.

More formally, suppose the training data is $\mathcal{D}_{train} = \{x_i, T(x_i), y_i\}_{i=1}^{N}$, where x_i is the original sample, y_i is the corresponding label, and $T(x_i) = \{t_{i1}, \cdots, t_{ic_i}\}$ is a collection of external knowledge descriptions for x_i. For a batch with s samples $x_1, \cdots, x_s$, the number of knowledge descriptions we need to encode is $|\bigcup_{i=1}^{s} T(x_i)|$.

For convenience, we assume that N is divisible by batch size s. To reduce the time complexity, we need to partition $\mathcal{D}_{train}$ into $k = N/s$ batches $B_1, \cdots, B_k$ such that each batch contains s samples and the total number of distinct textual commonsense descriptions in each batch is minimized:

$$\begin{aligned} \min \quad & \sum_{i=1}^{k} |\bigcup_{x \in B_i} T(x)| \\ \text{s.t.} \quad & |B_i| = s \ \text{ (size constraint for each batch)} \end{aligned} \tag{1}$$

3 Solving the Batch Partitioning Problem

To solve the batch partitioning problem, we first approximate the upper bound of Eq. (1) in § 3.1. We minimize its upper bound instead of directly minimizing Eq. (1). In § 3.2, we show that optimizing the upper bound can be reduced to the classic minimum graph k-cut problem, so that some well-studied algorithms can be applied. We show how we adapt the classical spectral clustering to this problem in § 3.3, and how to scale it up in § 3.4.

3.1 Upper Bound Analysis

We analyze the upper bound of Eq. (1) in Theorem 1.

Theorem 1 (Upper bound)**.**

$$\begin{aligned} & \sum_{i=1}^{k} |\bigcup_{x \in B_i} T(x)| \\ \leq & \sum_{i=1}^{k} [\sum_{x \in B_i} |T(x)| - s\, \mathbb{E}_{x_a, x_b \in B_i, x_a \neq x_b} |T(x_a) \cap T(x_b)|] \end{aligned} \tag{2}$$

Proof. For a batch B with s samples $\{x_1, \cdots, x_s\}$, we have:

$$\begin{aligned} & |\bigcup_{i=1}^{s} T(x_i)| = \sum_{i=1}^{s} |T(x_i) - \bigcup_{j=1}^{i-1} T(x_j)| \\ = & \sum_{i=1}^{s} |T(x_i) - \bigcup_{j=1}^{i-1} T(x_j) \cap T(x_i)| \\ = & \sum_{i=1}^{s} |T(x_i)| - \sum_{i=1}^{s} |\bigcup_{j=1}^{i-1} T(x_j) \cap T(x_i)| \\ \leq & \sum_{i=1}^{s} |T(x_i)| - \sum_{i=1}^{s} \max_{1 \leq j \leq i-1} |T(x_j) \cap T(x_i)| \end{aligned} \tag{3}$$

The upper bound in Eq. (3) after relaxation is related to the sample order of that batch, while our original objective in Eq. (1) is actually order-independent. To introduce order-independence, let π be an arrangement of $1 \cdots s$ that $\pi_i \in \{1, \cdots, s\}$. Noticing that $\sum_{i=1}^{s} |T(x_i)|$ is a constant, based on the order-independence, we transform Eq. (3) into the expectation under different πs:

$$\begin{aligned} & \mathbb{E}_\pi \sum_{i=1}^{s} \max_{1 \leq j \leq i-1} |T(x_{\pi_j}) \cap T(x_{\pi_i})| \\ = & \sum_{i=1}^{s} \mathbb{E}_\pi \max_{1 \leq j \leq i-1} |T(x_{\pi_j}) \cap T(x_{\pi_i})| \\ \geq & \sum_{i=1}^{s} \max_{1 \leq j \leq i-1} \mathbb{E}_\pi |T(x_{\pi_j}) \cap T(x_{\pi_i})| \\ = & \, s\, \mathbb{E}_{x_a, x_b \in B_i, x_a \neq x_b} |T(x_a) \cap T(x_b)| \end{aligned} \tag{4}$$

Therefore Theorem 1 holds. □

It is worth highlighting that the relaxation in the last inequality of Eq. (3) is valid due to the non-random distribution of words in samples. Specifically, samples with similar meanings tend to have similar word distributions. By grouping similar

samples into the same batch, each sample pair within a batch will possess similar textual commonsense knowledge descriptions. This allows us to use the maximal common descriptions between $T(x_i)$ and $T(x_j)$ as an approximation for the common descriptions between $T(x_i)$ and $\bigcup_{j=1}^{i-1} T(x_j)$.

According to Theorem 1, since $\sum_{i=1}^{s} \sum_{x \in B_i} |T(x)| = \sum_{x \in D_{train}} |T(x)|$ is a constant, minimizing Eq. (1) is equivalent to maximizing:

$$\sum_{i=1}^{k} \mathbb{E}_{x_a, x_b \in B_i, x_a \neq x_b} |T(x_a) \cap T(x_b)| \quad (5)$$

We will show that this is a balanced graph k-cut problem in § 3.2.

3.2 Connection to the Graph k-Cut Problem

We now illustrate the relationship between Eq. (5) and the graph k-cut problem. We demonstrate that, with proper transformation, maximizing Eq. (5) can be reduced to the graph k-cut problem. Additionally, in § 3.3, we explain how to incorporate the constraint of the size of each mini-batch using the balanced graph k-cut.

Consider constructing a weighted graph $G(V, E)$ as follows:

- For each sample x_i in the training data, create a vertex v_i.
- For each pair of distinct vertices (v_i, v_j), create an edge between them with a weight of $|T(x_i) \cap T(x_j)|$.

The graph k-cut for $G(V, E)$ partitions $G(V, E)$ into k non-empty components: $V_1, \cdots, V_k$ such that the sum weight of cross-component edges is minimized. According to the construction of $G(V, E)$, maximizing Eq. (5) is equivalent to minimizing the sum weight of the cut. This is formalized in Theorem 2.

Theorem 2 (Relation to minimum k-cut problem)**.** *Suppose the weight of the k-cut for $G(V, E)$ is w, then we have:*

$$Eq.\ (5) = \frac{2}{s(s-1)} \sum_{i=1}^{n-1} \sum_{j=i}^{n} |T(x_i) \cap T(x_j)| - w \quad (6)$$

Proof. A k-cut of $G(V, E)$ consists of k components. These k components correspond to k batches in the k-partition. Therefore, the sum weight of inner-component edges of the k-cut is equal to Eq. (5) $* \frac{s(s-1)}{2}$. Since the total weight of edges in $G(V, E)$ is equal to the sum weight of inner-component edges plus the sum weight of the cut, Theorem 2 holds. □

As $\sum_{i=1}^{n-1} \sum_{j=i}^{n} |T(x_i) \cap T(x_j)|$ is a constant for the given training data, Theorem 2 shows that maximizing Eq. (5) is equivalent to minimizing the k-cut for $G(V, E)$. Thus, we convert the problem of maximizing Eq. (5) into the classic minimum k-cut problem.

3.3 Spectral Clustering for the Balanced k-Cut

Based on the analysis in § 3.2, we propose to use spectral clustering, a widely used approach for solving the minimum graph k-cut problem, as our batch partition algorithm. Spectral clustering employs spectral relaxation of the ratio/normalized cut and uses k-means in the embedding of the vertices found by the first k eigenvectors of the graph Laplacian in order to obtain the clustering. In addition to the classic minimum graph k-cut problem, we need to incorporate the constraint that each cut/batch must have a size of s.

To incorporate the batch size constraint, we make a simple modification to the k-means step in spectral clustering. In the traditional k-means, each node is assigned to the nearest cluster center. In our algorithm, if the nearest cluster center has already been assigned s nodes, the node will be assigned to the nearest center that has fewer than s assigned nodes. The specific spectral clustering algorithm is presented as follows.

1. Compute the spectral embedding $Y \in \mathbb{R}^{n \times k}$ by stacking the normalized first k eigenvectors of $G(V, E)$ in columns as described in (Ng et al., 2002).
2. Treat the i-th row of Y as the feature of the i-th training point $e_i \in \mathbb{R}^k$.
3. Given an initial set of k means $m_1, \cdots, m_k$ by randomly selecting k nodes as centers, repeat the following two steps until convergence:
   (a) **Assignment step** Assign nodes to centers:
      i. Compute distances to centers $dis_{i,j} = distance(e_i, m_j)$, where the Euclidean distance is used.

ii. Sort i, j in ascending order of $dis_{i,j}$ for all $1 \leq i \leq n, 1 \leq j \leq k$.

iii. Iterate through all i, j. If node i is not assigned in this round and center j has less than s assigned nodes, assign node i to center j.

(b) **Update step** Compute new centers by taking the mean of their assigned nodes.

3.4 Spectral Clustering at Scale

The above algorithm consists of computation of the eigenvectors, and the use of k-means. K-means is efficient even for large-scale data. However, when n and k are large, the graph construction and eigenvectors computation become computationally expensive.

To compute the spectral embeddings at scale, high-performance optimization techniques are available such as (Liu et al., 2013; Kolev and Mehlhorn, 2016; Boutsidis et al., 2015; Tremblay et al., 2016). Also, in our experiments, a simple trick was found that yields meaningful results: only calculate k'-dimensional feature vectors ($k' < k$) and perform k-means with the k' dimensions. We found that $k' = 8$ is a good practice in our experiments.

4 Related Work

Integrating knowledge into language models has been one of the focuses of language modeling research in recent years. The main integration methods currently include using pre-trained entity embeddings, and constructing knowledge-aware corpora. ERNIE (Zhang et al., 2019), KnowBERT (Peters et al., 2019), and KGLM (Logan et al., 2019) are typical methods using pre-trained entity embeddings. ERNIE uses Wikidata (Vrandečić and Krötzsch, 2014) as the knowledge base and uses TransE (Bordes et al., 2013) to encode knowledge. KnowBERT, on the other hand, uses skip-gram like objective (Mikolov et al., 2013) based on Wikipedia descriptions as the pre-trained entity embeddings. In addition, KnowBERT adds a loss on entity linking to the pre-trained objective. KGLM (Logan et al., 2019) allows modification/updating of knowledge by building a local knowledge graph for the target sentence. WKLM (Xiong et al., 2019) constructs a corpus of incorrect knowledge descriptions by replacing Wikipedia's entities with different entities of the same type. It trains the model to identify incorrect and correct knowledge descriptions. Recently, models that integrate textual knowledge have also been proposed. In this paper, we adopt the model structure in OK-Transformer (Cui and Chen, 2022).

Textual knowledge bases Noting the deficiencies of symbolic knowledge in terms of expressiveness and contextual information representation, some work has started to use text as a form of knowledge. ATOMIC (Sap et al., 2019; Hwang et al., 2021) is a large-scale manually annotated common-sense textual knowledge base that includes social interaction, event-centered, physical entity. ATOMIC contains knowledge like *(PersonX reaches PersonX's home, Before, PersonX needs to park the car)*. ASER (Zhang et al., 2020) is an eventuality knowledge graph of activities, states, events, and their relations. Its knowledge atoms are in natural language form, e.g. *(I do not have lunch, succession, I am hungry)*. COMET (Bosselut et al., 2019) is an extension of ATOMIC based on the generative language model. It mainly solves the problem of insufficient coverage of ATOMIC. Some primitive research (Guan et al., 2020; Shwartz et al., 2020) has started to apply these textual knowledge bases in some specific tasks. OK-Transformer (Cui and Chen, 2022) is proposed to integrate textual knowledge for general purposes. However, in our experimental tests, it takes too much time in encoding the commonsense. To our knowledge, there is still a lack of research on how to integrate textual knowledge into general text understanding tasks efficiently.

Comparison with dense textual knowledge retriever When introducing external texts, another style is to use a retriever that returns only top k candidate texts in terms of similarity (Chen et al., 2017; Karpukhin et al., 2020; Wang et al., 2019). However, this method requires a heavy pre-training process to learn the retriever. On the other hand, for the textual knowledge base we use in this paper, we can directly use the manually labeled trigger words for each knowledge description to retrieve knowledge. Therefore, in this paper, we focus on how to efficiently and effectively integrate knowledge from a textual knowledge base.

High-performance language models More general techniques for high-performance language models have also received extensive studies. The main approaches of previous studies include (1) model compression and quantization (Sanh et al., 2019; Jacob et al., 2018), and (2) efficient repre-

sentation of long texts (Kitaev et al., 2019; Peng et al., 2020). However, the model compression approaches require heavy pre-training before they can be adapted to language models. Moreover, the techniques for optimizing the efficiency for long text do not have significant effects on short texts (Peng et al., 2020). Besides, each commonsense description we considered in this paper tends to be short. In addition, these works have not considered the characteristics of the knowledge integration problem in this paper, i.e., a training sample corresponds to multiple candidate textual knowledge from the knowledge base.

5 Experiments

In this section, we conducted extensive experiments to evaluate batch partitioning. We aim to address the following key questions:

1. (§ 5.2) How much is the efficiency improvement of batch partitioning? Can it improve efficiency without sacrificing effectiveness?

2. (§ 5.3) What is the scalability of batch partitioning as an acceleration method, and can it be applied to large-scale training?

3. (§ 5.4) Is the main theoretical contribution of this paper, i.e., solving the balanced graph-k cut by spectral clustering, consistent with the real datasets?

5.1 Implementation Details and Setup

Textual knowledge base We follow (Cui and Chen, 2022) to use ATOMIC2020 (Hwang et al., 2021) as the textual knowledge base. Each atom in ATOMIC2020 is commonsense in text form. For each sentence in the downstream task, we retrieve the knowledge associated with it from the textual knowledge base. Note that, unlike retrieving knowledge from free text (Guu et al., 2020), the textual knowledge base ATOMIC2020 is constructed manually, and each knowledge description has corresponding trigger words. These trigger words are usually verbs or verb phrases. We retrieve related textual commonsense descriptions by keyword-matching of these trigger words.

Model architecture We use OK-Transformer (Cui and Chen, 2022) as the backbone of our model. It directly incorporates extra knowledge without pre-training. OK-Transformer is based on either BERT or RoBERTa. We use OK-Transformer based on BERT by default. We also follow the hyperparameter settings of OK-Transformer. All experiments were run on 8 Nvidia RTX 3090Ti GPUs.

Datasets We evaluate batch partitioning via commonsense reasoning and sentence classification. Since the textual knowledge introduced in this paper is commonsense descriptions, we first verify whether the proposed method in this paper could be applied to the commonsense reasoning tasks. To this end, we choose a wide range of commonsense reasoning tasks to conduct the experiments: CommonsenseQA (Talmor et al., 2019), PhysicalQA (Bisk et al., 2020), as well as several Winograd Schema Challenge (WSC) datasets including WSC273 (Levesque et al., 2012), PDP (Morgenstern et al., 2016), WinoGrande (Sakaguchi et al., 2019), WinoGender (Rudinger et al., 2018). Furthermore, for a comprehensive comparison, we also evaluate the efficiency and effectiveness of the proposed batch partitioning method on the text classification benchmark GLUE (Wang et al., 2018).

5.2 Effectiveness and Efficiency

Baselines To verify the efficiency and effectiveness of batch partitioning, we used the following baselines:

- **Vanilla BERT/RoBERTa** without external knowledge.

- **OK-Transformer** To show the efficiency gains of the batch partitioning proposed in this paper, we compare it with the original OK-Transformer. The baseline randomly partitions samples into batches. We consider this baseline as **the lower upper bound of effectiveness** of commonsense integration.

- **Frozen knowledge encodings** For a comprehensive comparison, we propose to freeze the encoding of commonsense descriptions during fine-tuning. This approach allows us to introduce external textual commonsense descriptions via embedding lookup with minimal time cost. We consider this baseline as **the upper bound on the efficiency** of commonsense integration.

The results of commonsense reasoning and text classification are presented in Table 2 and Table 3, respectively. The effectiveness of our batch partitioning approach is demonstrated by its improvement over vanilla language models on both commonsense reasoning and text classification tasks.

	LM	Comm.QA	PhysicalQA	WSC273	PDP	WinoGrande	WinoGender	Avg.	Speed-up ↑
BERT	BERT	55.86	68.71	66.30	85.00	51.38	68.19	65.44	-
Frozen knowledge	BERT	56.43	68.06	65.93	83.33	51.30	68.47	65.59	**1.4**×
OK-Transformer	BERT	56.27	69.09	**67.40**	**86.67**	**52.64**	71.53	66.56	1.0×
Batch Partitioning	BERT	**56.59**	**69.53**	66.67	**86.67**	52.17	**72.78**	**67.40**	**1.4**×
RoBERTa	RoB.	73.55	79.76	90.10	90.00	-	94.60	83.95	-
Frozen knowledge	RoB.	75.02	52.77	90.48	88.33	-	**96.81**	80.01	**1.5**×
OK-Transformer	RoB.	**75.92**	80.09	**91.58**	90.00	-	95.00	84.75	1.0×
Batch Partitioning	RoB.	75.59	**80.20**	90.48	**91.66**	-	96.25	**85.14**	1.4×

Table 2: Results on commonsense reasoning tasks. The effectiveness of batch partitioning surpasses the vanilla BERT/RoBERTa, and is competitive with its upper bound (OK-Transformer). In terms of efficiency, the speed-up of batch partitioning is also competitive to its upper bound (frozen knowledge). RoB. denotes RoBERTa.

	LM	MRPC	CoLA	RTE	QNLI	STS-B	SST-2	Avg.	Speed-up
BERT	BERT	86.52/90.66	59.50	71.43	91.20	89.35/88.93	91.97	82.28	-
Frozen knowledge	BERT	87.50/91.28	57.31	70.76	91.71	87.31/87.20	92.43	81.78	**2.3**×
OK-Transformer	BERT	87.50/91.04	58.29	**72.20**	**91.58**	**89.82/89.46**	92.66	82.54	1.0×
Batch Partitioning	BERT	**87.99/91.45**	**61.41**	71.48	91.32	89.64/89.19	**93.69**	**83.09**	2.1×
RoBERTa	RoB.	90.49/93.07	66.84	86.28	93.37	91.83/91.95	95.64	87.86	-
Frozen knowledge	RoB.	89.71/92.61	**68.22**	**87.36**	94.39	90.74/90.47	96.10	88.19	**2.4**×
OK-Transformer	RoB.	**91.91/94.24**	66.89	86.28	**94.71**	92.19/92.36	**96.44**	**88.49**	1.0×
Batch Partitioning	RoB.	90.69/93.44	67.75	85.92	94.07	**92.41/92.20**	96.22	88.27	2.1×

Table 3: Results on text classification tasks. Both the effectiveness and the efficiency of batch partitioning are competitive to their upper bounds (OK-Transformer and frozen knowledge).

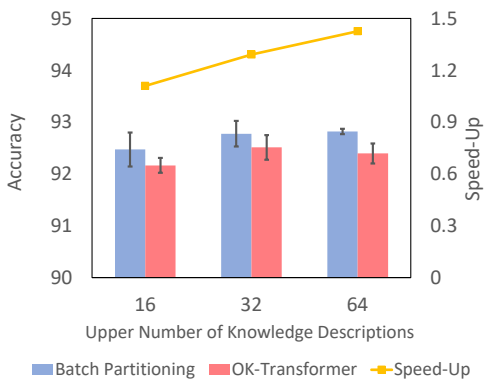


Figure 2: The effect of the scale of extra commonsense. We control the scale by limiting the upper number of commonsense descriptions per sample in SST-2.

The effectiveness is comparable or slightly superior to that of OK-Transformer, which serves as the upper bound for effectiveness. In terms of efficiency, our approach significantly accelerates knowledge integration models across a range of tasks. On average, it reduces the time cost for knowledge encoding by 40% for commonsense reasoning tasks, and 110% for text classification tasks. This acceleration is close to the frozen knowledge, and serves as the upper bound for efficiency. Overall, our approach is close to its efficiency upper bound without losing effectiveness.

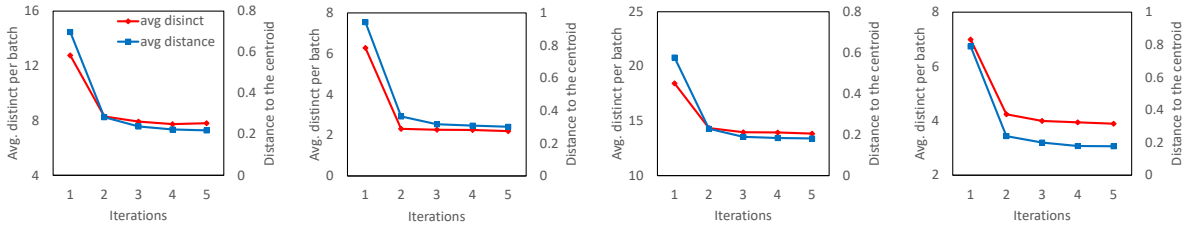
5.3 Scalability for Dataset Sizes, Device Capacities, and Knowledge Sizes

In this subsection, we investigate the scalability of batch partitioning with different batch sizes, as well as the different dataset sizes. Larger dataset sizes usually mean devices with larger memory. In particular, we calculated the speedups of knowledge encoding for different batch sizes and different tasks. The results are shown in Fig. 4. The datasets are sorted by size in descending order.

It can be clearly seen that as the size of the dataset rises or the memory of the device rises (larger batch size), the speedup of batch partitioning becomes more significant. This is because, for data-intensive tasks, the knowledge overlapping among different samples is more significant, which increases the feasibility of using batch partitioning. This result verifies the scalability of batch partitioning.

We also investigate the scalability of batch partitioning over different scales of integrated commonsense. To control the scale, we set the upper number of commonsense descriptions for each sample to 16/32/64, respectively, and study the efficiency. Intuitively, richer commonsense descriptions lead to higher effectiveness but more computation cost. The results are shown in Fig. 2.

As commonsense knowledge becomes richer, the



(a) MRPC (b) CoLA (c) RTE (d) STS-B

Figure 3: Strong correlation between the distinct commonsense descriptions per batch and the average distance to the centroid during clustering.

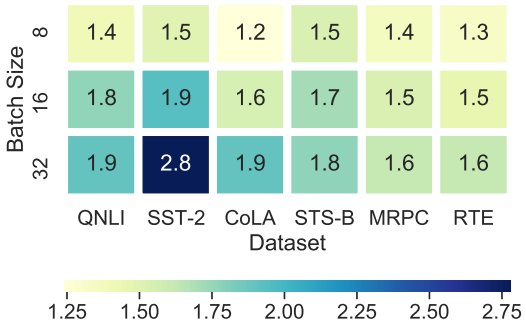


Figure 4: Speed-up of proposed batch partitioning method over different training batch size and dataset size. Note that the datasets are arranged in descending order according to dataset size.

effectiveness and the acceleration both increase. This is because the knowledge overlapping among samples also becomes more significant. The result verifies that batch partitioning is applicable for incorporating large-scale commonsense knowledge bases.

5.4 Effect of Spectral Clustering Theory

In this paper, we propose the use of spectral clustering to solve the batch partitioning problem. We approximate and optimize the distinct number of descriptions per batch in Eq. (1) by minimizing the distance of each node to the centroid of the cluster in spectral clustering. In this subsection, we demonstrate the rationale behind this approximation by highlighting the strong correlation between the objective of Eq. (1) and the distance minimization in spectral embeddings.

To this end, we plot how the centroid distance and the distinct descriptions per batch vary at each iteration of the spectral clustering algorithm in Fig. 3. The results show a strong correlation between the value we directly optimize (i.e., the centroid distance) and the target of the batch partitioning (i.e., distinct descriptions per batch). This supports the feasibility of using spectral clustering to convert the batch partitioning problem into a balanced graph k-cut problem and solve it efficiently.

6 Conclusion

In this paper, we study how to improve the efficiency of incorporating commonsense knowledge in language models. Due to the high encoding costs of commonsense descriptions, it is crucial to reduce their encoding complexity. Our idea is that by carefully dividing samples with similar descriptions into the same batch, the knowledge encoding utilization can be improved.

With such an idea, we theoretically analyze the optimization objective of this batch partitioning. We found that the upper bound of this problem can be reduced to the classical graph k-cut problem. We propose to use the well-studied spectral clustering algorithm to optimize the batch partitioning. By experimenting with a variety of tasks, we show that the proposed batch partitioning approaches its upper bound in terms of both effectiveness and efficiency. And the method is more applicable for larger datasets and on devices with more capabilities.

7 Limitations

The theoretical results and the algorithm should be applicable for other knowledge integration models which encode target sentences and associated textual knowledge descriptions in mini-batches. However, this paper does not extensively apply the proposed method to various knowledge integration models to explore its efficiency and effectiveness.

References

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7432–7439.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, Florence, Italy. Association for Computational Linguistics.

Christos Boutsidis, Prabhanjan Kambadur, and Alex Gittens. 2015. Spectral clustering via the power method-provably. In *International conference on machine learning*, pages 40–48. PMLR.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Wanyun Cui and Xingran Chen. 2022. Enhancing natural language representation with large-scale out-of-domain commonsense. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1746–1756, Dublin, Ireland. Association for Computational Linguistics.

Tianqing Fang, Hongming Zhang, Weiqi Wang, Yangqiu Song, and Bin He. 2021. Discos: Bridging the gap between discourse knowledge and commonsense knowledge. In *Proceedings of the Web Conference 2021*, pages 2648–2659.

Jian Guan, Fei Huang, Zhihao Zhao, Xiaoyan Zhu, and Minlie Huang. 2020. A knowledge-enhanced pretraining model for commonsense story generation. *Transactions of the Association for Computational Linguistics*, 8:93–108.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International Conference on Machine Learning*, pages 3929–3938. PMLR.

Jena D. Hwang, Chandra Bhagavatula, Ronan Le Bras, Jeff Da, Keisuke Sakaguchi, Antoine Bosselut, and Yejin Choi. 2021. Comet-atomic 2020: On symbolic and neural commonsense knowledge graphs. In *AAAI*.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2019. Reformer: The efficient transformer. In *International Conference on Learning Representations*.

Pavel Kolev and Kurt Mehlhorn. 2016. A note on spectral clustering. In *24th Annual European Symposium on Algorithms (ESA 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*. Citeseer.

Jialu Liu, Chi Wang, Marina Danilevsky, and Jiawei Han. 2013. Large-scale spectral clustering on graphs. In *Twenty-Third International Joint Conference on Artificial Intelligence*.

Robert Logan, Nelson F. Liu, Matthew E. Peters, Matt Gardner, and Sameer Singh. 2019. Barack's wife hillary: Using knowledge graphs for fact-aware language modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5962–5971, Florence, Italy. Association for Computational Linguistics.

Chaitanya Malaviya, Chandra Bhagavatula, Antoine Bosselut, and Yejin Choi. 2020. Commonsense knowledge base completion with structural and semantic context. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2925–2933.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Leora Morgenstern, Ernest Davis, and Charles L Ortiz. 2016. Planning, executing, and evaluating the winograd schema challenge. *AI Magazine*, 37(1):50–54.

Andrew Y Ng, Michael I Jordan, and Yair Weiss. 2002. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong. 2020. Random feature attention. In *International Conference on Learning Representations*.

Matthew E Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A Smith. 2019. Knowledge enhanced contextual word representations. In *Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Syama Sundar Rangapuram, Pramod Kaushik Mudrakarta, and Matthias Hein. 2014. Tight continuous relaxation of the balanced k-cut problem. *Advances in Neural Information Processing Systems*, 27:3131–3139.

Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. 2018. Gender bias in coreference resolution. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 8–14.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A Smith, and Yejin Choi. 2019. Atomic: An atlas of machine commonsense for if-then reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3027–3035.

Vered Shwartz, Peter West, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Unsupervised commonsense question answering with self-talk. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4615–4629.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158.

Nicolas Tremblay, Gilles Puy, Rémi Gribonval, and Pierre Vandergheynst. 2016. Compressive spectral clustering. In *International conference on machine learning*, pages 1002–1011. PMLR.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang. 2019. Multi-passage bert: A globally normalized bert model for open-domain question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5878–5882.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6397–6407.

Wenhan Xiong, Jingfei Du, William Yang Wang, and Veselin Stoyanov. 2019. Pretrained encyclopedia: Weakly supervised knowledge-pretrained language model. In *International Conference on Learning Representations*.

Hongming Zhang, Xin Liu, Haojie Pan, Yangqiu Song, and Cane Wing-Ki Leung. 2020. Aser: A large-scale eventuality knowledge graph. In *Proceedings of the web conference 2020*, pages 201–211.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. Ernie: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Sec 5

☐ A2. Did you discuss any potential risks of your work?
Not applicable. Left blank.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Left blank.

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☒ Did you use or create scientific artifacts?

Left blank.

☐ B1. Did you cite the creators of artifacts you used?
Not applicable. Left blank.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Not applicable. Left blank.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Not applicable. Left blank.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Not applicable. Left blank.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Left blank.

C ☑ Did you run computational experiments?

we discuss the experiments in sec5

☒ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
parameters same as BERT/RoBERTa.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
in sec5

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Fig. 2

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
We follow the implementation and experiments setting of OK-Transformer as we mentioned in sec 5.

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Not applicable. Left blank.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Not applicable. Left blank.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Not applicable. Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.