CONTEXTUALIZED SPARSE REPRESENTATION WITH RECTIFIED N-GRAM ATTENTION FOR OPEN-DOMAIN QUESTION ANSWERING

Anonymous authors
Paper under double-blind review

ABSTRACT

A sparse representation is known to be an effective means to encode precise lexical cues in information retrieval tasks by associating each dimension with a unique n-gram-based feature. However, it has often relied on term frequency (such as tf-idf and BM25) or hand-engineered features that are coarse-grained (document-level) and often task-specific, hence not easily generalizable and not appropriate for fine-grained (word or phrase-level) retrieval. In this work, we propose an effective method for learning a highly contextualized, word-level sparse representation by utilizing rectified self-attention weights on the neighboring n-grams. We *kernelize* the inner product space during training for memory efficiency without the explicit mapping of the large sparse vectors. We particularly focus on the application of our model to phrase retrieval problem, which has recently shown to be a promising direction for open-domain question answering (QA) and requires lexically sensitive phrase encoding. We demonstrate the effectiveness of the learned sparse representations by not only drastically improving the phrase retrieval accuracy (by more than 4%), but also outperforming all other (pipeline-based) open-domain QA methods with up to x97 faster inference in $\text{SQUAD}_{\text{OPEN}}$ and CURATEDTREC.[1]

1 INTRODUCTION

Retrieving text from a large collection of documents is an important problem in several natural language tasks such as question answering (QA) and information retrieval. In the literature, sparse representations have been successfully used in encoding text at sentence or document level, capturing precise lexical information that can be sparsely activated by n-gram based features. For instance, frequency-based sparse representations such as tf-idf map each text segment to a vocabulary space where the weight of each dimension is determined by the associated word's term and inverse document frequency. However, these sparse representations are mainly coarse-grained, task-specific, and not suitable for word-level representations since they statically assign the identical weight to each n-gram and do not change dynamically depending on the context.

In this paper, we introduce an effective method for learning a word-level sparse representation that encodes precise lexical information. Our model, COSPR, learns a *co*ntextualized *s*parse *p*hrase *r*epresentation leveraging rectified self-attention weights on the neighboring n-grams and dynamically encoding important lexical information of each phrase (such as named entities) given its context. Consequently, in contrast to previous sparse regularization techniques on dense embedding of at most few thousand dimensions (Faruqui et al., 2015; Subramanian et al., 2018), our method is able to produce more interpretable representations with billion-scale cardinality. Such large sparse vector space is prohibitive for explicit mapping; we leverage the fact that our sparse representations only interact through inner product and *kernelize* the inner-product space for memory-efficient training, inspired by the kernel method in SVMs (Cortes & Vapnik, 1995). This allows us to handle extremely large sparse vectors without worrying about computational bottlenecks. The overview of our model is illustrated in Figure 1.

We demonstrate the effectiveness of our model in open-domain question answering (QA), the task of retrieving answer phrases given a web-scale collection of documents. Following Seo et al. (2019),

[1] Our code will be publicly available upon acceptance.

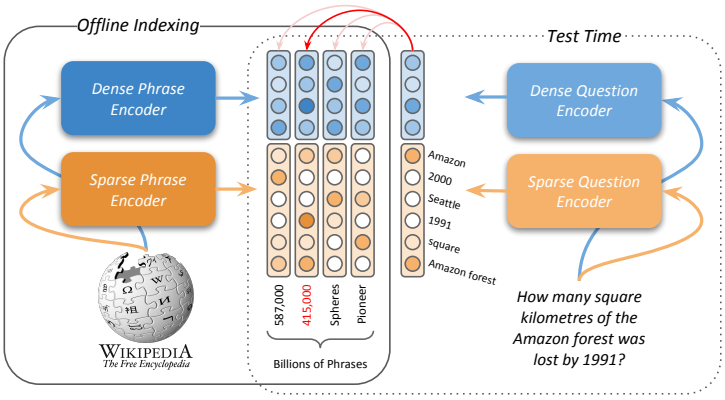

Figure 1: Overview of our model. Using the billions of precomputed phrase representations, we perform a maximum inner product search between the phrase vectors and an input question vector. We propose to learn contextualized sparse phrase representations which are also very interpretable.

we concatenate both sparse and dense vectors to encode every phrase in Wikipedia and use maximum similarity search to find the closest candidate phrase to answer each question. We only substitute (or augment) the baseline sparse encoding which is entirely based on frequency-based embedding (tf-idf) with our contextualized sparse representation (CoSPR). Our empirical results demonstrate its state-of-the-art performance in open-domain QA datasets, SQuAD$_{\text{OPEN}}$ and CuratedTrec. Notably, our method significantly outperforms DenSPI (Seo et al., 2019), the previous end-to-end QA model, by more than 4% with negligible drop in inference speed. Moreover, our method achieves up to 2% better accuracy and x97 speedup in inference compared to pipeline (retrieval-based) approaches. Our analysis particularly shows that fine-grained sparse representation is crucial for doing well in phrase retrieval task. In summary, the contributions of our paper are:

1. we show that learning sparse representations for embedding lexically important context words of a phrase can be achieved by contextualized sparse representations,
2. we introduce an efficient training strategy that leverages the *kernelization* of the sparse inner-product space, and
3. we achieve the state-of-the-art performance in two open-domain QA datasets with at least x97 faster inference time.

2 Related Work

Open-domain question answering Most open-domain QA models for unstructured texts use a retriever to find documents to read, and then apply a reading comprehension (RC) model to find answers (Chen et al., 2017; Wang et al., 2018a; Lin et al., 2018; Das et al., 2019; Lee et al., 2019; Yang et al., 2019; Wang et al., 2019). To improve the performance of the open-domain question answering, various modifications have been studied to the pipelined models which include improving the retriever-reader interaction (Wang et al., 2018a; Das et al., 2019), re-ranking paragraphs and/or answers (Wang et al., 2018b; Lin et al., 2018; Lee et al., 2018; Kratzwald et al., 2019), learning end-to-end models with weak supervision (Lee et al., 2019), or simply making a better retriever and a reader model (Yang et al., 2019; Wang et al., 2019). Due to the pipeline nature, however, these models inevitably suffer error propagation from the retrievers.

To mitigate this problem, Seo et al. (2019) propose to learn query-agnostic representations of phrases in Wikipedia and retrieve phrases that best answers a question. While Seo et al. (2019) have shown that encoding both dense and sparse representations for each phrase could keep the lexically important words of a phrase to some extent, their sparse representations are based on static tf-idf vectors which have globally the same weight for each n-gram.

Phrase representations In NLP, phrase representations can be either obtained in a similar manner as word representations (Mikolov et al., 2013), or by learning a parametric function of word representations (Cho et al., 2014). In extractive question answering, phrases are often referred to as spans, but most models do not consider explicitly learning phrase representations as these answer spans can be obtained by predicting only start and end positions in a paragraph (Wang & Jiang, 2017; Seo et al., 2017). Nevertheless, few studies have focused on directly learning and classifying phrase representations (Lee et al., 2017) which achieve strong performance when combined with attention mechanism. In this work, we are interested in learning query-agnostic sparse phrase representations which enables the precomputation of re-usable phrase representations (Seo et al., 2018).

Sparse representations One of the most basic sparse representations is the *bag-of-words* modeling, and recent works often emphasize the use of bag-of-words models as strong baselines for sentence classification and question answering (Joulin et al., 2017; Weissenborn et al., 2017). *tf-idf* is another good example of sparse representations that is used for document retrieval, and is still widely adopted both in IR and QA community. Our work could be seen as an attempt to build a trainable tf-idf model for phrase (n-gram) representations, which should be more fine-grained than paragraphs or documents. There have been some attempts in IR that learn sparse representations of documents for duplicate detection (Hajishirzi et al., 2010), inverted indexing (Zamani et al., 2018), and more. Unlike previous works, however, our method does not require hand-engineered features for sparse n-grams while keeping the original vocabulary space. In NLP, there are some works on training sparse representations specifically designed for improving interpretability of word representations (Faruqui et al., 2015; Subramanian et al., 2018), but they lose an important role of sparse represenations, which is keeping the exact lexical information, as they are trained by sparsifying dense representations of a higher (but much lower than V) dimension.

3 Background: Open-Domain QA through Phrase Retrieval

Open-Domain QA We primarily focus on open-domain QA on unstructured data where the answer is a text span in the corpus. Formally, given a set of K documents $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^K$ and a question $\boldsymbol{q}$, the task is to design a model that obtains the answer $\hat{\boldsymbol{a}}$ by

$$\hat{\boldsymbol{a}} = \arg\max_{\boldsymbol{x}^k_{i:j}} F(\boldsymbol{x}^k_{i:j}, \boldsymbol{q}),$$

where F is the score model to learn and $\boldsymbol{x}^k_{i:j}$ is a phrase consisting of words from the i-th to the j-th word in the k-th document. Typically, the number of documents (K) is in the order of millions for open-domain QA (e.g. 5 million for English Wikipedia), which makes the task computationally challenging. Pipeline-based methods typically leverage a document retriever to reduce the number of documents to read, but they suffer from error propagation when wrong documents are retrieved and can be slow if the reader model is computationally cumbersome.

Open-domain QA with Phrase Encoding and Retrieval As an alternative, phrase-retrieval approaches (Seo et al., 2018; 2019) mitigate this problem by directly accessing all phrases in K documents by decomposing F into two functions,

$$\hat{a} = \arg\max_{\boldsymbol{x}^k_{i:j}} F(\boldsymbol{x}^k_{i:j}, \boldsymbol{q}) = \arg\max_{\boldsymbol{x}^k_{i:j}} H_{\boldsymbol{x}}(\boldsymbol{x}^k_{i:j}) \cdot H_{\boldsymbol{q}}(\boldsymbol{q})$$

where $\cdot$ denotes inner product operation. Unlike running a complex reading comprehension model in pipeline-based approaches, $H_{\boldsymbol{x}}$ query-agnostically encode (all possible phrases of) each document just once, so that we just need to compute $H_{\boldsymbol{q}}$ (which is very fast) and perform similarity search on the phrase encoding (which is also fast). Seo et al. (2019) have shown that encoding each phrase with a concatenation of dense and sparse representations is effective, where the dense part is computed from BERT (Devlin et al., 2019) and the sparse part is obtained from the tf-idf vector of the document and the paragraph of the phrase. We briefly describe how the dense part is obtained below.

Dense Representation Assuming that the document $\boldsymbol{x}$ has N words as $x_1, \ldots, x_N$, Seo et al. (2019) use BERT (Devlin et al., 2019) to compute contextualized representation of each word as $\mathbf{h}_1, \ldots, \mathbf{h}_N = \text{BERT}(x_1, \ldots, x_N)$. Based on the contextualized embeddings, we obtain dense

phrase representations as follows: We split each $\mathbf{h}_i \in \mathbb{R}^d$ into four parts $[\mathbf{h}_i^1, \mathbf{h}_i^2, \mathbf{h}_i^3, \mathbf{h}_i^4]$ where $\mathbf{h}_i^1, \mathbf{h}_i^2 \in \mathbb{R}^{d_{se}}$ and $\mathbf{h}_i^3, \mathbf{h}_i^4 \in \mathbb{R}^{d_c}$ (hence $d = 2d_{se} + 2d_c$). Then each phrase $\boldsymbol{x}_{i:j}$ is densely represented as follows:

$$\mathbf{h}_{i:j} = [\mathbf{h}_i^1, \mathbf{h}_j^2, \mathbf{h}_i^3 \cdot \mathbf{h}_j^4] \in \mathbb{R}^{2d_{se}+1} \tag{1}$$

where $\cdot$ denotes inner product operation. $\mathbf{h}_i^1$ and $\mathbf{h}_j^2$ are start/end representations of a phrase, and the inner product of $\mathbf{h}_i^3$ and $\mathbf{h}_j^4$ is used for computing coherency of the phrase. Refer to Seo et al. (2019) for details; we mostly reuse its architecture.

4 Sparse Encoding of Phrases

Sparse representations are often suitable for keeping the precise lexical information present in the text, complementing dense representations that are often good for encoding semantic and syntactic information. While the sparsification of dense word embeddings has been explored previously (Faruqui et al., 2015), its main limitations are that (1) it starts from the dense embedding which might have already lost rich lexical information, and (2) its cardinality is in the order of thousands at max, which we hypothesize is too small to encode sufficient information. Our work hence focuses on creating the sparse representation of each phrase which is not bottlenecked by dense embedding and is capable of increasing the cardinality to billion-scale without much computational cost.

4.1 Why do we need to learn sparse representations?

Suppose we are given a question "*How many square kilometres of the Amazon forest was lost by 1991?*" and the target answer is in the following sentence,

> *Between 1991 and 2000, the total area of forest lost in the Amazon rose from **415,000** to 587,000 square kilometres (160,000 to 227,000 sq mi).*

To answer the question, the model should know that the target answer (***415,000***) corresponds to the year *1991* while the (confusing) phrase *587,000* corresponds to the year *2000*, which requires syntactic understanding of English parallelism. The dense phrase encoding is likely to have a difficulty in precisely differentiating between *1991* and *2000* since it needs to also encode several different kinds of information. Window-based tf-idf would not help either because the year *2000* is closer (in word distance) to ***415,000***. This example illustrates the strong need to create an n-gram-based sparse encoding that is highly syntax- and context-aware.

4.2 Contextualized Sparse Representations

Our sparse model, unlike pre-computed sparse embeddings such as tf-idf, *dynamically* computes the weight of each n-gram that depends on the context. Intuitively, for each phrase, we want to compute a positive weight for each n-gram near the phrase depending on how important the n-gram is to the phrase.

Sparse Representation Sparse representation of each phrase is also obtained as the concatenation of its start word's and end word's sparse embedding, i.e. $\mathbf{s}_{i:j} = [\mathbf{s}_i^{\text{start}}, \mathbf{s}_j^{\text{end}}]$. This way, similarly to how the dense phrase embedding is obtained, we can efficiently compute them without explicitly enumerating all possible phrases.

We obtain each (start and end) sparse embedding in the same way (with unshared parameters), so we just describe how we obtain the start sparse embedding here and omit the superscript 'start'. Given the contextualized encoding of each document $\mathbf{H} = [\mathbf{h}_1, \ldots, \mathbf{h}_N] \in \mathbb{R}^{N \times d}$, we obtain its (start or end) sparse encoding $\mathbf{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_N] \in \mathbb{R}^{N \times F}$ by

$$\mathbf{S} = \text{ReLU}\big(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\big)\mathbf{F} \in \mathbb{R}^{N \times F} \tag{2}$$

where $\mathbf{Q}, \mathbf{K} \in \mathbb{R}^{N \times d}$ are query, key matrices obtained by applying a (different) linear transformation on $\mathbf{H}$ (i.e., using $W_\mathbf{Q}, W_\mathbf{K} : \mathbb{R}^{N \times d} \to \mathbb{R}^{N \times d}$), and $\mathbf{F} \in \mathbb{R}^{N \times F}$ is an one-hot n-gram feature

representation of the input document $\boldsymbol{x}$. That is, for instance, if we want to encode unigram (1-gram) features, $\mathbf{F}_i$ will be simply a one-hot representation of the word x_i, and F will be equivalent to the vocabulary size. Note that F will be very large, so it should always exists as an efficient sparse matrix format (e.g. csc) and one should not explicitly create its dense form. We also discuss how we can leverage it during training in an efficient manner in Section 4.3.

Note that Equation 2 is similar to the scaled dot-product self-attention (Vaswani et al., 2017), with two key differences that (1) ReLU instead of softmax is used, and (2) a sparse matrix $\mathbf{F}$ instead of (dense) value matrix is used. In fact, our sparse embedding is related to attention mechanism in that we want to compute how *important* each n-gram is for each phrase's start and end word. Intuitively, $\mathbf{s}_i$ contains a weighted bag-of-ngram representation where each n-gram is weighted by its relative importance on each start or end word of a phrase. Unlike attention mechanism whose role is to summarize the target vectors via weighted summation (thus softmax is used, which sums up to 1), we do not perform the summation and directly output the unnormalized attention weights to a large sparse embedding space. In fact, we experimentally observe that ReLU is more effective than softmax for this objective.

Since we want to handle several different sizes of n-grams, we create the sparse encoding $\mathbf{S}$ for each n-gram and concatenate the resulting sparse encodings. That is, let $\mathbf{S}^n$ be the sparse encoding for n-gram, then the final sparse encoding is the concatenation of different n-grams we consider. In practice, we experimentally find that unigram and bigram are sufficient for most use cases; in this case, the sparse vector for the (start) word $\boldsymbol{x}_i$ will be $\mathbf{s}_i = [\mathbf{s}_i^1, \mathbf{s}_i^2]$. We do not share linear transformation parameters for across different n-grams. Note that this is also analogous to multiple attention heads in Vaswani et al. (2017).

Question Encoding For a question $\boldsymbol{q} = \texttt{[CLS]}, \ldots, q_M$ where `[CLS]` denotes a special token for BERT inputs, contextualized question representations are computed in a similar way ($\mathbf{h}'_{\texttt{[CLS]}}, \ldots \mathbf{h}'_M = \text{BERT}(\texttt{[CLS]}, \ldots, q_M)$). We share the same BERT used for the phrase encoding. We compute sparse encodings on the question side ($\mathbf{s}' \in \mathbb{R}^F$) in a similar way to the document side, with the only difference that we use the `[CLS]` token instead of start and end words to represent the entire question. Linear transformation weights are not shared.

4.3 Training

Kernel function As training phrase encoders on the whole Wikipedia is computationally prohibitive, we use training examples from an extractive question answering dataset (SQuAD) to train our encoders. Given a pair of question $\boldsymbol{q}$ and a golden document $\boldsymbol{x}$ (a paragraph in the case of SQuAD), we first compute the dense logit of each phrase $\boldsymbol{x}_{i:j}$ by $l_{i,j} = \mathbf{h}_{i:j} \cdot \mathbf{h}'$.

Unlike Seo et al. (2019), each phrase's sparse embedding is also trained, so it needs to be considered in the loss function. We define the sparse logit for phrase $\boldsymbol{x}_{i:j}$ as $l^{\text{sparse}}_{i,j} = \mathbf{s}_{i:j} \cdot \mathbf{s}'_{\texttt{[CLS]}} = \mathbf{s}^{\text{start}}_i \cdot \mathbf{s}'^{\text{start}}_{\texttt{[CLS]}} + \mathbf{s}^{\text{end}}_j \cdot \mathbf{s}'^{\text{end}}_{\texttt{[CLS]}}$. For brevity, we describe how we compute the first term $\mathbf{s}^{\text{start}}_i \cdot \mathbf{s}'^{\text{start}}_{\texttt{[CLS]}}$ corresponding to the start word (and dropping the superscript 'start'); the second term can be computed in the same way.

$$\mathbf{s}^{\text{start}}_i \cdot \mathbf{s}'^{\text{start}}_{\texttt{[CLS]}} = \text{ReLU}\Big(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\Big)_i \mathbf{F} \left(\text{ReLU}\Big(\frac{\mathbf{Q}'\mathbf{K}'^\top}{\sqrt{d}}\Big)_{\texttt{[CLS]}} \mathbf{F}' \right)^\top$$

where $\mathbf{Q}', \mathbf{K}' \in \mathbb{R}^{M \times d}, \mathbf{F}' \in \mathbb{R}^{M \times F}$ denote the question side query, key, and n-gram feature matrices, respectively. We can efficiently compute it if we precompute $\mathbf{F}\mathbf{F}'^\top \in \mathbb{R}^{N \times M}$. Note that $\mathbf{F}\mathbf{F}'^\top$ can be considered as applying a kernel function, i.e. $K(\mathbf{F}, \mathbf{F}') = \mathbf{F}\mathbf{F}'^\top$ where its (i,j)-th entry is 1 if and only if n-gram at i-th position of the context is equivalent to j-th n-gram of the question, which can be efficiently computed as well. One can also think of this as *kernel trick* (in the literature of SVM (Cortes & Vapnik, 1995)) that allows us to compute the loss function without explicit mapping.

The loss to minimize is computed from the negative log likelihood over the sum of the dense and sparse logits:

$$L = -(l_{i^*,j^*} + l^{\text{sparse}}_{i^*,j^*}) + \log \sum_{i,j} \exp(l_{i,j} + l^{\text{sparse}}_{i,j}) \tag{3}$$

where i^*, j^* denote the true start and end positions of the answer phrase. While the loss above is an unbiased estimator, in practice, we adopt early loss summation as suggested by Seo et al. (2019) for larger gradient signals in early training. Additionally, we also add dense-only loss that omits the sparse logits (i.e. original loss in Seo et al. (2019)) to the final loss, in which case we find that we obtain higher-quality dense phrase representations.

Negative sampling We train our model on SQuAD v1.1 which always has a positive paragraph that contains the answer. To learn robust phrase representations, we concatenate negative paragraphs to the original SQuAD paragraphs. To each paragraph $\boldsymbol{x}$, we concatenate the paragraph $\boldsymbol{x}_{\text{neg}}$ which was paired with the question whose dense representation $\mathbf{h}'_{\text{neg}}$ is most similar to the original dense question representation $\mathbf{h}'$, following Seo et al. (2019). Note the difference, however, that we *concatenate* the negative example instead of considering it as an independent example with no-answer option Levy et al. (2017). During training, we find that adding tf-idf matching scores on the word-level logits of the negative paragraphs further improves the quality of sparse representations as our sparse models have to give stronger attentions to positively related words in this biased setting.

5 EXPERIMENTS

5.1 DATASETS

We use two popular open-domain QA datasets to evaluate our model: $\text{SQUAD}_{\text{OPEN}}$ and CURATEDTREC. $\text{SQUAD}_{\text{OPEN}}$ is the open-domain version of SQUAD (Rajpurkar et al., 2016). We use 87,599 examples with the golden evidence paragraph to train our encoders, and use 10,570 QA pairs from development set to test our model, as suggested by Chen et al. (2017). CURATEDTREC consists of question-answer pairs from TREC QA (Voorhees et al., 1999) constructed based on real user queries. We use 694 test set QA pairs for testing our model. Note that we only train on SQuAD and test on both SQuAD and CuratedTREC, relying on the generalization ability of our model for zero-shot inference on CuratedTREC. This is in a clear contrast to previous work that utilize distant supervision (Chen et al., 2017) or weak supervision (Lee et al., 2018; Min et al., 2019) on CuratedTREC.

5.2 IMPLEMENTATION DETAILS

We use and finetune $\text{BERT}_{\text{LARGE}}$ for our encoders. We use BERT vocabulary which has 30522 unique tokens based on byte pair encodings. As a result, we have $F = 30522$ when using unigram feature for $\mathbf{F}$, and $F \approx 1B$ when using both uni/bigram features. We do not finetune the word embedding during training. We pre-compute and store all encoded phrase representations of all documents in Wikipedia (more than 5 million documents). It takes 600 GPU hours to index all phrases in Wikipedia. Each phrase representation has $2d_{se} + 2F + 1$ dimensions. We use the same storage reduction and search techniques by Seo et al. (2019). For storage, the total size of the index is 1.3 TB including unigram and bigram sparse representations. For search, we either perform dense search first and then rerank with sparse scores (DFS) or perform sparse search first and rerank with dense scores (SFS), and also consider a combination of both (Hybrid).

Constraining sparse encoding We find that injecting some prior domain knowledge to avoid spurious matchings allows us to learn better sparse representations. For example, in machine reading comprehension, answer phrases do not occur as an exact phrase in questions (e.g., the phrase "*August 4, 1961*" would not appear in the question "*When was Barack Obama born?*"). Therefore, we can zero the elements in the diagonal axis of $\mathbf{Q}\mathbf{K}^\top \in \mathbb{R}^{N \times N}$ in Equation 2 which correspond to the attention values of target phrase itself. Additionally, we mask special tokens in BERT such as `[SEP]` or `[CLS]` to have zero weights as matching of these tokens means nothing.

5.3 RESULTS

We evaluate the effectiveness of COSPR by augmenting DENSPI (Seo et al., 2019) with contextualized sparse representations (DENSPI+COSPR). We extensively compare the model with the original DENSPI and previous pipeline-based QA models.

Model	SQuAD$_{\text{OPEN}}$		CuratedTrec	
	EM	F1	Exact Match	s/Q
DrQA (Chen et al., 2017)	29.8**	-	25.4*	35
R^3 (Wang et al., 2018a)	29.1	37.5	28.4*	-
Paragraph Ranker (Lee et al., 2018)	30.2	-	35.4*	-
Multi-Step-Reasoner (Das et al., 2019)	31.9	39.2	-	-
BERTserini (Yang et al., 2019)	38.6	46.1	-	115
ORQA (Lee et al., 2019)	20.2	-	30.1	-
Multi-passage BERT†† (Wang et al., 2019)	53.0	60.9	-	-
DenSPI (Seo et al., 2019)	36.2	44.4	31.6†	0.81
DenSPI+CoSPR(Ours)	**40.7**	**49.0**	**35.7**†	1.19

* Trained on distantly supervised training data.
** Trained on multiple datasets
† No supervision using target training data.
†† Concurrent work.

Table 1: Results on open-domain question answering datasets.

Model	SQuAD$_{1/100}$	SQuAD$_{1/10}$
Ours (DFS)	60.0	51.6
− CoSPR	55.9 (−4.1)	48.4 (−3.2)
− doc./para. tf-idf	58.1 (−1.9)	50.9 (−0.7)
− tf-idf bias training	58.1 (−1.9)	49.1 (−2.5)
+ trigram sparse	58.0 (−2.0)	49.8 (−1.8)

Model	SQuAD$_{\text{OPEN}}$		CuratedTrec	
	DenSPI	+ CoSPR	DenSPI	+ CoSPR
SFS	33.3	36.9 (+3.6)	28.8	30.0 (+1.2)
DFS	28.5	34.4 (+5.9)	29.5	34.3 (+4.8)
Hybrid	36.2	40.7 (+4.5)	31.6	35.7 (+4.1)

Table 2: Ablations of our model. On the left, we show how each sparse representation contribute to the performance, and on the right, we show how much gain we obtain from CoSPR in different search strategies. Exact match scores are reported.

Open-domain QA experiments Table 1 shows experimental results on two open-domain question answering datasets. For DenSPI and ours, we use Hybrid search strategy. On both datasets, our model with contextualized sparse representations (DenSPI + CoSPR) significantly improves the performance of the phrase-indexing baseline model (DenSPI). Moreover, our model outperforms BERT-based pipeline approaches such as BERTserini (Yang et al., 2019) while being almost two orders of magnitude faster. We expect much bigger speed gaps between ours and other pipeline methods as most of them put additional complex components to the original pipelined methods.

On CuratedTrec, which is constructed from real user queries, our model also achieves the state-of-the-art performance. Even though our model is only trained on SQuAD (i.e. zero-shot), it outperforms all other models which are either distant- or semi-supervised with at least 29x faster inference. Note that our method is orthogonal to end-to-end training (Lee et al., 2019) or weak supervision (Min et al., 2019) methods and future work can potentially benefit from these.

Ablation study Table 2 (left) shows the effect of contextualized sparse representations by comparing different variants of our method on SQuAD$_{\text{OPEN}}$. For these evaluations, we use a subset of Wikipedia dump (1/100, 1/10 scale, approximately 50K, 500K documents, respectively). We evaluate the effect of removing (1) CoSPR, (2) tf-idf sparse representations, (3) tf-idf bias on negative sample training (section 4.3), and (4) and adding trigram features to our sparse representation (i.e., $\mathbf{s}_i = [\mathbf{s}_i^1, \mathbf{s}_i^2, \mathbf{s}_i^3]$). One interesting observation is that adding trigram features in our sparse representations is worse than using uni-/bigram representations only. We suspect that the model becomes too dependent on trigram features, which means we might need a stronger regularization for high-order n-gram features.

In Table 2 (right), we show how we consistently improve over DenSPI when CoSPR is added in different search strategies. Note that on SQuAD$_{\text{OPEN}}$, SFS is better than DFS as questions in SQuAD are created by turkers given particular paragraphs, whereas on CuratedTrec where the questions more resemble real user queries, DFS outperforms SFS showing the effectiveness of dense search when not knowing which documents to read. Similar observation was pointed by Lee et al. (2019) who showed different performance tendencies between two datasets, but as we are using

Context (**phrase**), Question	Sparse rep.	*Top n-gram* (weight)
Context: Between 1991 and 2000, the total area of forest lost in the Amazon rose from **415,000** to 587,000 square kilometres.	tf-idf	*amazon rose* (9.65), . . . , *1991* (2.26), *2000* (1.85)
	CoSPR	*1991* (1.65), *amazon* (0.82), . . . , *2000* (0.28)
Context: Amongst these include 5 University of California campuses (. . .); **12** California State University campuses (. . .);	tf-idf	*12 california* (8.80), *5 university* (8.04)
	CoSPR	*california state* (1.94), *state university* (1.77)
Question: Who did Newton get a pass to in the Panther starting plays of Super Bowl 50?	tf-idf	*starting plays* (9.31), *panther starting* (9.14)
	CoSPR	*50* (2.20), *newton* (1.62), *super bowl* (1.52)

Table 3: Analysis of CoSPR compared to static tf-idf vectors. Given a context or a question (left), we list top n-grams weighted by tf-idf or CoSPR (right).

Q: When is Independence Day?	
DrQA	[*Independence Day (1996 film)*] Independence Day is a **1996** American science fiction ...
DenSPI+CoSPR	[*Independence Day: Resurgence*] It is the sequel to the **1996** film "Independence Day".
	[*Independence Day (India)*] ... is annually observed on **15 August** as a national holiday in India.
	[*Independence Day (Pakistan)*] (...), observed annually on **14 August**, is a national holiday ...
Q: What was the GDP of South Korea in 1950?	
DrQA	[*Economy of South Korea*] In 1980, the South Korean GDP per capita was **$2,300**.
DenSPI	[*Economy of South Korea*] In 1980, the South Korean GDP per capita was **$2,300**.
DenSPI + CoSPR	[*Developmental State*] South Korea's GDP grew from **$876** in 1950 to $22,151 in 2010.

Table 4: Prediction samples from DrQA, DenSPI, and DenSPI+CoSPR. Each sample shows [*document title*], context, and **predicted answer**.

both dense and sparse representations, our model can achieve state-of-the-art performances on both datasets.

5.4 Analysis

Interpretability of sparse representations Sparse representations often have better interpretability than dense representations as each dimension of a sparse vector corresponds to a specific word. We compare tf-idf vectors and CoSPR (uni/bigram) by showing top weighted n-grams in each representation. Note that the scale of weights in tf-idf vectors is normalized in open-domain setups to match the scale between tf-idf vectors and dense vectors. We observe that tf-idf vectors usually assign high weights on infrequent (often meaningless) n-grams, while CoSPR focuses on contextually important entities such as *1991* for *415,000* or *californio state*, *state university* for *12*. Our sparse question representation also learns meaningful n-gram weights compared to tf-idf vectors.

Prediction samples Table 4 shows the outputs of three OpenQA models: DrQA (Chen et al., 2017), DenSPI (Seo et al., 2019), and our DenSPI+CoSPR. DenSPI+CoSPR is able to retrieve various correct answers from different documents, and it often correctly answers questions with specific dates or numbers compared to DenSPI showing the effectiveness of learned sparse representations.

6 Conclusion

In this paper, we demonstrate the effectiveness of contextualized sparse vectors, CoSPR, for encoding phrase with rich lexical information in open-domain question answering. We efficiently train our sparse representations by kernelizing the sparse inner product space. Experimental results show that our fast open-domain QA model that augments the previous model (DenSPI) with learned sparse representation (CoSPR) outperforms previous open-domain QA models, including recent BERT-based pipeline models, with two orders of magnitude faster inference time. Future work includes extending the contextualized sparse representations to other challenging QA settings such as multi-hop reasoning and building a full inverted index of the learned sparse representations (Zamani et al., 2018) for more powerful Sparse-First Search (SFS).

REFERENCES

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. In *ACL*, 2017.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 1995.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. Multi-step retriever-reader interaction for scalable open-domain question answering. In *ICLR*, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A Smith. Sparse overcomplete word vector representations. In *ACL-IJCNLP*, 2015.

Hannaneh Hajishirzi, Wen-tau Yih, and Aleksander Kolcz. Adaptive near-duplicate detection via similarity learning. In *SIGIR*, 2010.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In *ACL*, 2017.

Bernhard Kratzwald, Anna Eigenmann, and Stefan Feuerriegel. Rankqa: Neural question answering with answer re-ranking. In *ACL*, 2019.

Jinhyuk Lee, Seongjun Yun, Hyunjae Kim, Miyoung Ko, and Jaewoo Kang. Ranking paragraphs for improving answer recall in open-domain question answering. In *EMNLP*, 2018.

Kenton Lee, Shimi Salant, Tom Kwiatkowski, Ankur Parikh, Dipanjan Das, and Jonathan Berant. Learning recurrent span representations for extractive question answering. In *ICLR*, 2017.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised open domain question answering. In *ACL*, 2019.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. In *CoNLL*, 2017.

Yankai Lin, Haozhe Ji, Zhiyuan Liu, and Maosong Sun. Denoising distantly supervised open-domain question answering. In *ACL*, 2018.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

Sewon Min, Danqi Chen, Hannaneh Hajishirzi, and Luke Zettlemoyer. A discrete hard em approach for weakly supervised question answering. In *EMNLP*, 2019.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *ACL*, 2016.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. In *ICLR*, 2017.

Minjoon Seo, Tom Kwiatkowski, Ankur Parikh, Ali Farhadi, and Hannaneh Hajishirzi. Phrase-indexed question answering: A new challenge for scalable document comprehension. In *EMNLP*, 2018.

Minjoon Seo, Jinhyuk Lee, Tom Kwiatkowski, Ankur P Parikh, Ali Farhadi, and Hannaneh Hajishirzi. Real-time open-domain question answering with dense-sparse phrase index. In *ACL*, 2019.

Anant Subramanian, Danish Pruthi, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Eduard Hovy. Spine: Sparse interpretable neural embeddings. In *AAAI*, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

Ellen M Voorhees et al. The trec-8 question answering track report. In *Trec*, 1999.

Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. In *ICLR*, 2017.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerry Tesauro, Bowen Zhou, and Jing Jiang. R 3: Reinforced ranker-reader for open-domain question answering. In *AAAI*, 2018a.

Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. Evidence aggregation for answer re-ranking in open-domain question answering. In *ICLR*, 2018b.

Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang. Multi-passage bert: A globally normalized bert model for open-domain question answering. In *EMNLP*, 2019.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. Making neural qa as simple as possible but not simpler. In *CoNLL*, 2017.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. End-to-end open-domain question answering with bertserini. In *NAACL Demo*, 2019.

Hamed Zamani, Mostafa Dehghani, W Bruce Croft, Erik Learned-Miller, and Jaap Kamps. From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In *CIKM*, 2018.