CSurF: Sparse Lexical Retrieval through Contextualized Surface Forms

Zhen Fan
Carnegie Mellon University
Pittsburgh, PA, USA
zhenfan@andrew.cmu.edu

Luyu Gao
Carnegie Mellon University
Pittsburgh, PA, USA
luyug@cs.cmu.edu

Jamie Callan
Carnegie Mellon University
Pittsburgh, PA, USA
callan@cs.cmu.edu

ABSTRACT

Lexical exact-match systems perform text retrieval efficiently with sparse matching signals and fast retrieval through inverted lists, but naturally suffer from the mismatch between lexical surface form and implicit term semantics. This paper proposes to directly bridge the surface form space and the term semantics space in lexical exact-match retrieval via *contextualized surface forms* (CSF). Each CSF pairs a lexical surface form with a context source, and is represented by a lexical form weight and a contextualized semantic vector representation. This framework is able to perform sparse lexicon-based retrieval by learning to represent each query and document as a "bag-of-CSFs", simultaneously addressing two key factors in sparse retrieval: vocabulary expansion of surface form and semantic representation of term meaning. At retrieval time, it efficiently matches CSFs through exact-match of learned surface forms, and effectively scores each CSF pair via contextual semantic representations, leading to joint improvement in both term match and term scoring. Multiple experiments show that this approach successfully resolves the main mismatch issues in lexical exact-match retrieval and outperforms state-of-the-art lexical exact-match systems, reaching comparable accuracy as lexical all-to-all soft match systems as an efficient exact-match-based system.

CCS CONCEPTS

• **Information systems** → *Document representation*; *Query representation*; **Retrieval models and ranking**; *Search engine architectures and scalability*.

KEYWORDS

Sparse retrieval, lexical exact match, contextualized surface forms, deep language models

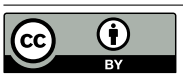
ACM Reference Format:
Zhen Fan, Luyu Gao, and Jamie Callan. 2023. CSurF: Sparse Lexical Retrieval through Contextualized Surface Forms. In *Proceedings of the 2023 ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR '23), July 23, 2023, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3578337.3605126

1 INTRODUCTION

Lexical exact match, or matching query and document based on overlapping terms, has been widely utilized in text retrieval [37]. The strict and imperfect premise to indifferently match all terms with identical lexical surface form (e.g. matching all documents containing the word "Christmas" or "present" to the query "Christmas present") leads to sparse text sequence representations and matching signals, which enables fast retrieval via inverted indexing. However, this premise naturally neglects any information about the term semantics beneath a surface form, leading to vocabulary mismatch (same meaning shared by multiple surface forms) and semantic mismatch (different meanings under the same surface form) and consequently suboptimal retrieval performance.

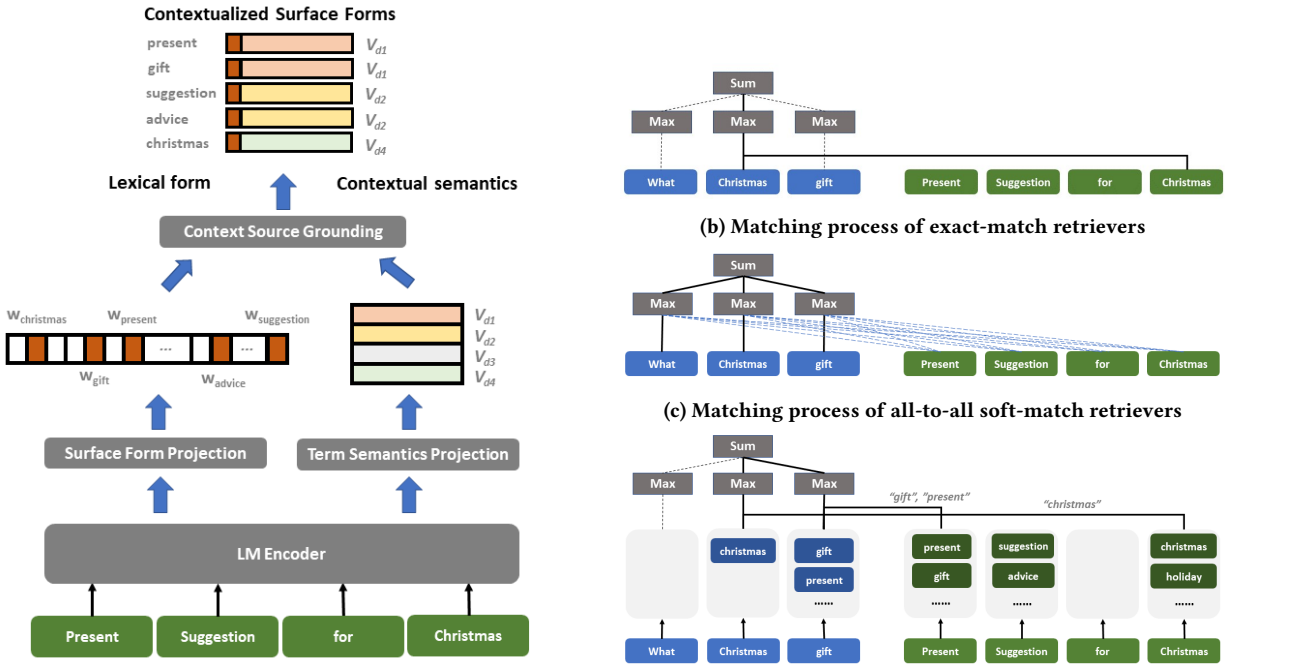
The era of neural IR and pretrained language models [8, 42] has led to several directions of work to train end-to-end lexical matching retrievers. Within the term-weighting premise, SPLADE [11, 12] and its numerous extension models perform implicit vocabulary expansion, projecting the original text sequence to a new set of surface forms with learned term weights, but do not further track or distinguish the individual semantics of the generated lexical forms. Other systems propose to augment lexical form matching with contextualized vector representations and use vector similarity scoring to distinguish semantic agreement between query and document terms. Gao et al. [15] proposed *contextualized inverted lists* (COIL) to reduce semantic mismatch within the lexical exact-match framework, but this approach does not change the vocabulary of the text sequence and thus suffers from vocabulary mismatch. ColBERT [21, 39] systems further perform soft *all-to-all match*, completely removing the lexical form match restriction and matching all possible query-doc term pairs with vector representation scoring. This results in maximum model capacity but sacrifices the efficiency gain of the exact-match framework, leading to impractical indexing and storage cost for first-stage retrieval.

The advances and limitations of current systems inspire us to rethink the lexical text retrieval process. Lexical retrievers judge a query-document pair by *matching* query and document terms and *scoring* each term pair. In the matching stage, performing exact lexical match via surface form naturally ensures sparse signals, which is fundamental for retrieval efficiency. In the scoring stage, matching terms by semantic representations produces more precise term match scores, but this step itself does not determine the correctness and sparsity of term matching signals. Previous systems utilizing vector term representations still match terms through the lexical surface forms of the original text and thus suffer from vocabulary mismatch of exact match (COIL) or efficiency issues of all-to-all soft match (ColBERT). On the other hand, SPLADE-based systems demonstrate the potential to learn new "bag-of-surface-forms" to overcome the mismatch in query-document vocabulary, but such learned surface forms are disconnected from the original context. An ideal end-to-end system should combine the advantage of the



ICTIR '23, July 23, 2023, Taipei, Taiwan

ACM ISBN 979-8-4007-0073-6/23/07.
https://doi.org/10.1145/3578337.3605126



(a) Model structure and encoding process of CSurF.

(b) Matching process of exact-match retrievers

(c) Matching process of all-to-all soft-match retrievers

(d) Matching process of CSurF

Figure 1: The model structure of the CSurF retriever (left), and the retrieval process of CSurF compared to current lexicon-based exact match or soft match retrievers(right). CSurF generates Contextualized Surface Forms (CSFs) by pairing lexical forms with semantic representations of original text terms. CSFs are further indexed in inverted lists and matched based on lexical form.

two sides, to learn to jointly find term matches via surface form, and further score such matches via term semantic representation.

This paper proposes CSurF, which enables effective sparse retrieval through **C**ontextualized **Sur**face **F**orms. Each contextualized surface form consists of a lexical surface form with an importance weight and a context source with a contextualized representation. We use CSurF to name the retrieval model, and CSF to name the "contextualized surface form" concept, which is the basic unit of the sparse lexical match process. Figure 1a demonstrates the model structure of the CSurF framework, and examples of generated CSFs. The model represents each query and document using a "bag-of-CSFs" by first generating a set of lexical surface forms. It grounds all such surface forms to a contextual semantics source in the original text to assign each surface form with a vector representation. At retrieval time, the CSurF model first sparsely matches CSFs by exact-match of lexical surface forms, and further scores each CSF pair by comparing contextual representation similarity, which assures both the efficiency and effectiveness of the retrieval process.

Figure 1b-1d further demonstrate the retrieval process of CSurF compared to lexical exact-match and all-to-all soft-match systems. CSurF is able to link the *original* query term "gift" with the document term "present" since both the query and document generate CSFs with surface form "gift" and "present". From the classic lexical exact-match perspective, CSurF simultaneously incorporates lexical form expansion and contextualized scoring to resolve vocabulary and semantic mismatch in an end-to-end system. From the lexical soft-match perspective, CSurF efficiently connects original query-doc terms through fast exact-match of projected surface forms, leading to improved efficiency without loss in model capacity. CSurF outperforms current lexical exact-match systems on multiple in-domain and zero-shot retrieval experiments, consistently reaching the performance level of state-of-the-art lexical all-to-all soft-match models, but in an efficient sparse retrieval framework. We further discuss the effectiveness-efficiency tradeoff of CSurF and its improvements over current lexical retrieval systems, including the benefits of context source grounding, vector term representation extension, and sparse lexical soft match.

The rest of the paper is structured as follows: Section 2 discusses related work on neural lexical retrieval systems. Section 3 introduces the CSurF framework. Sections 4 and 5 discuss experiments and analysis of model performance. Section 6 concludes.

2 RELATED WORK

Before the emergence of pretrained language models, classic text retrieval systems [37] rely on exact-match signals of query and document terms to provide relevance judgment. Such systems assign each term with an importance weight estimated from term frequency, and perform efficient retrieval via inverted indexing. Traditional approaches to resolve the vocabulary and semantic mismatch in lexical exact-match systems include rewriting and expansion of the query and document [1, 3, 4, 43, 47], as well as

n-gram matching [26]. Later systems utilize pretrained word embeddings such as Word2vec [27] to calculate term similarity [17], and propose lexical *soft match* or all-to-all match [7, 45], performing complete interaction between query and document terms regardless of lexical form, but such systems are thus much less efficient and often limited to re-ranking settings.

The introduction of BERT-scale pretrained language models [8, 42] provides a natural backbone for text understanding, and has led to significant improvement in both re-ranking [30] and first-stage ranking. Pretrained LMs were first used to improve BM25-based retrieval via improved term-reweighting [6] or document expansion [31, 32]. Later research proposed to directly learn query and document representations for retrieval. One thread of work involves building *dense retrievers* to directly encode the query and document into a single vector representation, and directly predict relevance based on vector similarity. Main research in dense retrieval focuses on better training strategies such as negative sampling [46, 48], knowledge distillation [18, 25] and training objective design [13, 14, 16, 19], as well as retrieval efficiency [20, 49].

This paper mainly focuses on end-to-end *lexicon-based retrieval* or *learned sparse retrieval* [2, 24, 28], which represents the query and document with a set of vocabulary terms and term representations, and predicts relevance from aggregating term-level shallow interactions. Some systems follow the exact-match precondition and aim to perform efficient retrieval with contextualized term importance weights [12, 24] or semantic vector representations [15, 50]. To resolve vocabulary mismatch, systems such as SPLADE [12] and its extensions [11] also introduce the concept of vocabulary expansion to remap the original bag-of-words to a new set of vocabulary terms. Such systems further benefit from the speed-up of the inverted list structure to store the scalar or vector term representations [15]. Another group of LM-augmented lexicon-based systems performs lexical soft match with all-to-all token interaction and vector scoring, with the most important being ColBERT [21, 39]. Soft-match retrievers fall on the other side of the capacity-efficiency tradeoff with high retrieval performance but also more token interactions required. Recent advances accordingly focus on reducing its storage and computational costs with approaches including residual compression and centroid pruning [38, 39]. In this paper, we aim to find a direct approach to both *allow* and *control* the soft interaction between original terms with different surface forms, achieving accurate retrieval with sparse matching signals.

Concurrent with our work, Li et al. [23] proposed CITADEL which utilizes token-level vocabulary expansion as "routing" to achieve controlled soft match of original text terms with contextualized vector representations. While this approach is similar to our CSurF approach, the expanded tokens are solely generated by term-wise top-k selection of expansion tokens, and rely on post-hoc pruning of generated expansion tokens to balance retrieval efficiency. This is contrary to CSurF which directly and dynamically learns a set of surface forms for the input text sequence. Qian et al. [35] proposed ALIGNER, which directly learns sparse alignment between query and document terms by predicting token salience, or whether a given token requires alignment with other tokens. This leads to direct pruning of the all-to-all soft match matrix of ColBERT, and thus improved retrieval efficiency.

3 CONTEXTUALIZED SURFACE FORMS

In this section, we first formally define the lexical match and retrieval process in Section 3.1. Sections 3.2 and 3.3 introduce the encoding and retrieval process of CSurF respectively. Section 3.4 discusses the connection and advances of CSurF compared to current systems. Section 3.5 discusses model implementation and training.

3.1 Prelimiaries

Given a query $\mathbf{q} = \{q_1, q_2, \cdots, q_m\}$ and document $\mathbf{d} = \{d_1, d_2, ..., d_n\}$, a lexical retriever scores the $(\mathbf{q}, \mathbf{d})$ pair by accumulating individual term match scores, commonly via the "max-sum" framework:

$$Score(\mathbf{q}, \mathbf{d}) = \sum_{q_i \in \mathbf{q}} \max_{d_j \in \mathbf{d}} \mathcal{M}(q_i, d_j) \mathcal{S}(q_i, d_j)$$

For query term q_i and document term d_j, $\mathcal{S}(q_i, d_j)$ is a **term scoring operation**, such as term weight multiplication or vector similarity calculation. $\mathcal{M}(q_i, d_j)$ is the *term matching criteria*, indicating whether q_i and d_j are matched. The strictness of the selection mask $\mathcal{M}$ directly determines the capacity and efficiency of the lexical retriever. As shown in Figure 1b, in exact-match systems, $\mathcal{M}(q_i, d_j) = \mathbb{I}(q_i = d_j)$ and only terms with identical form are matched. This results in sparse matching signals but suffers from the mismatch in query and document vocabulary. On the other hand, for all-to-all soft-match systems such as ColBERT, $\mathcal{M}(q_i, d_j) \equiv \mathbf{1}$ and all possible term pairs are matched regardless of surface form. This leads to maximum model capacity but at the cost of extremely high computation and storage cost.

3.2 Contextualized Surface Form Generation

CSurF aims to jointly control the sparsity of matching signals $\mathcal{M}$ and the precision of the scoring function $\mathcal{S}$ through sparse retrieval of **contextualized surface forms** (CSF). For each input sequence (query or document), it generates a bag-of-CSFs by (1) generating candidate lexical surface forms for the sequence, and (2) pairing each generated surface form with a term in the original text as its context source.

CSurF follows Formal et al. [11] and generates candidate lexical surface forms from the entire vocabulary space. It first encodes each input sequence (query or document) with a language model (LM) backbone and separately projects the LM output of each query and document term to two components: a dense representation denoting term semantics, and a sparse $|V|$-dim vector denoting expansion weights for each token in the vocabulary.

$$\mathbf{v}_{q_i} = \phi_v(LM(q, i)) \quad \mathbf{E}_{q_i} = \phi_e(LM(q, i))$$
$$\mathbf{v}_{d_j} = \phi_v(LM(d, j)) \quad \mathbf{E}_{d_j} = \phi_e(LM(d, j))$$

For query term q_i, $LM(q, i)$ denotes its LM output representation. ϕ_v and ϕ_e denote the semantic and expansion projection layers, which we discuss in Section 3.5. $\mathbf{v}_{q_i} \in \mathbb{R}^{|d_{rep}|}$ denotes the contextual semantics representation for q_i. $\mathbf{E}_{q_i} \in \mathbb{R}^{|V|}$ represents non-negative expansion weights from q_i to each token in the vocabulary V. For instance, $\mathbf{E}_{q_i[t]}$ represents the expansion weight from original term q_i to surface form $t \in V$, with higher weight value denoting higher likelihood of expansion. Query and document term representations are processed through the same projection layers.

Over the whole query or document sequence, for surface form t, CSurF performs max-pooling over all expansion weights $\mathbf{E}_{q[t]}$ to select its sequence importance weight. Additionally, CSurF **grounds** the surface form t to the original text by tracking the **projection source** of the selected expansion weight.

$$w_t^q = \max_{q_i \in q} \mathbf{E}_{q_i[t]} \quad s_t^q = \arg\max_{q_i \in q} \mathbf{E}_{q_i[t]}$$
$$w_t^d = \max_{d_j \in d} \mathbf{E}_{d_j[t]} \quad s_t^d = \arg\max_{d_j \in d} \mathbf{E}_{d_j[t]}$$

Given a query q and surface form t, w_t^q and s_t^q respectively denote the final expansion weight and its projection source, i.e. $E_{s_t,t} = w_t$.

The grounding step crucially links the lexical surface form space and the original text semantic space, enabling CSurF to pair each surface form with a context source and construct CSFs. We denote a contextualized surface form as $A = (t_A, s_A)$, where $t_A \in V$ is its lexical surface form with a scalar importance weight w_{t_A}, and $s_A \in q$ or $s_A \in d$ is its context source with semantic representation $\mathbf{v}_{s_A}$. This enables CSurF to combine the advantage of lexical form matching and semantic-based scoring, which we discuss in the following sections.

For each query and document, the expansion-based "bag-of-CSFs" $\mathcal{E}$ is the set of all CSFs with positive surface form weights.

$$\mathcal{E}^q = \{(t, s_t^q) \mid t \in V, w_t^q > 0\}$$
$$\mathcal{E}^d = \{(t, s_t^d) \mid t \in V, w_t^d > 0\}$$

Additionally, we construct CSFs for all *original* query and document terms to preserve the lexical form information of the original text. We directly use the term's self-projection weight $E^q_{q_i,q_i}$ or $E^d_{d_j,d_j}$ as its lexical form weight.

$$O^q = \{(q_i, q_i) \mid q_i \in \mathbf{q}, E^q_{q_i[q_i]} > 0\}$$
$$O^d = \{(d_j, d_j) \mid d_j \in \mathbf{d}, E^d_{d_j[d_j]} > 0\}$$

where O denotes the original text-based CSFs. The final bag-of-CSFs C for each query and document is the union of expansion-based and original text-based CSFs.

$$C^q = O^q \cup \mathcal{E}^q \quad C^d = O^d \cup \mathcal{E}^d$$

3.3 Indexing and Retrieval

CSurF scores a query-document pair by sparse retrieval through their bag-of-CSFs C^q and C^d. Specifically, it strictly matches CSF pairs through lexical exact match of their surface forms, and scores a matched CSF pair through vector similarity. For CSFs $A = (t_A, s_A) \in C^q$ and $B = (t_B, s_B) \in C^d$,

$$\begin{aligned}\mathcal{M}(A,B) &= \mathbb{I}(t_A = t_B)\\ \mathcal{S}(A,B) &= w_{t_A} w_{t_B}\, \mathbf{f}(\mathbf{v}_{s_A}, \mathbf{v}_{s_B})\\ Score(A,B) &= \mathcal{M}(A,B)\, \mathcal{S}(A,B)\\ &= \mathbb{I}(t_A = t_B) w_{t_A} w_{t_B}\, \mathbf{f}(\mathbf{v}_{s_A}, \mathbf{v}_{s_B})\end{aligned}$$

Here, $\mathbf{f}()$ denotes a similarity function such as dot-product or cosine similarity. CSurF jointly utilizes the efficiency of lexical form match and the accuracy of contextualized term scoring. The exact-match precondition $\mathcal{M}$ limits the density of actual CSF matches, which enables the model to efficiently index document CSFs in **inverted lists** via their lexical surface forms. The contextualized scoring component further complements lexical match by introducing term semantics.

CSurF follows the common "max-sum" framework to aggregate individual CSF match scores. At retrieval time, for each *original* query term, it selects the maximum score over all document CSFs.

$$\begin{aligned}Score(q_i, \mathbf{d}) &= \max_{A \in C_i^{\mathbf{q}}} \max_{B \in C^{\mathbf{d}}} Score(A,B)\\ &= \max_{A \in C_i^{\mathbf{q}}} \max_{B \in C^{\mathbf{d}}} \mathcal{M}(A,B)\, \mathcal{S}(A,B)\\ &= \max_{A \in C_i^{\mathbf{q}}} \max_{B \in C^{\mathbf{d}}} \mathbb{I}(t_A = t_B)\, w_{t_A} w_{t_B}\, \mathbf{f}(\mathbf{v}_{s_A}, \mathbf{v}_{s_B})\end{aligned}$$

where $C_i^{\mathbf{q}}$ denotes the subset of CSFs with q_i as source term. The final (q, d) score is the sum of individual query term scores.

$$\begin{aligned}Score(\mathbf{q}, \mathbf{d}) &= \sum_{q_i \in \mathbf{q}} Score(q_i, \mathbf{d})\\ &= \sum_{q_i \in \mathbf{q}} \max_{A \in C_i^{\mathbf{q}}} \max_{B \in C^{\mathbf{d}}} \mathcal{M}(A,B) \mathcal{S}(A,B)\\ &= \sum_{q_i \in \mathbf{q}} \max_{A \in C_i^{\mathbf{q}}} \max_{B \in C^{\mathbf{d}}} \mathbb{I}(t_A = t_B)\, w_{t_A} w_{t_B}\, \mathbf{f}(\mathbf{v}_{s_A}, \mathbf{v}_{s_B})\end{aligned}$$

3.4 Connection to Current Systems

CSurF is a direct extension of current lexicon-based retrieval models. From the surface form exact-match perspective, CSurF maintains the exact-match precondition and supports inverted indexing and sparse retrieval of contextualized surface forms. It simultaneously addresses the vocabulary and semantic mismatch issues of lexical exact-match by performing surface form expansion and assigning a contextualized representation to each surface form. The lexical exact-match systems COIL-tok and SPLADE can both be viewed as special cases of CSurF with limitations. COIL-tok is equivalent to CSurF with only original text-based surface forms $O^{\mathbf{q}}$ and $O^{\mathbf{d}}$ and without expansion, thus suffering in model capacity. SPLADE can be viewed as CSurF matching surface forms with only term weights ($|\mathbf{v}| = 0$) and with slightly different score accumulation methods. Specifically, the CSurF name framework enables SPLADE-based systems to be expanded to utilize vector term representations, by explicitly grounding each expanded surface form to the original context. We analyze the performance of CSurF compared to baseline lexical retrievers in Sections 5.1 and 5.3.

From the contextualized semantic match perspective, CSurF performs multi-vector lexical retrieval over the vector representations of query and document terms in the original text, but further introduces the surface form space to efficiently bridge the terms. CSurF is the equivalent of ColBERT with a strong surface form match constraint. It maps each term in the original text sequences to a set of candidate surface forms, so that original terms can be matched via expansion surface form overlap. The model is trained to jointly *learn* the soft-match constraint $\mathcal{M}$ of whether terms should match, along with the vector scoring $\mathcal{S}$, to filter unnecessary soft matches. This dramatically reduces the computation and indexing overload of ColBERT systems and leads to significantly improved efficiency, which we discuss in Section 5.2.

Table 1: Retrieval results on MSMARCO. Best performance under each training setting is labeled in bold.

Model		MSMARCO Dev		TREC 19	
Retriever	Initialization	MRR@10	R@1000	NDCG@10	R@1000
Models w/o distillation and hard negative mining					
BM25	-	0.187	0.857	0.506	0.745
DocT5Query + BM25	-	0.277	0.947	0.642	0.827
COIL-tok	BERT	0.341	0.949	0.660	-
COIL-tok	CoCondenser	0.353	0.949	0.692	0.801
SPLADE [12]	BERT	0.322	0.955	0.665	0.813
SPLADE [11]	BERT	0.342	0.966	0.699	0.815
CoCondenser	-	0.357	**0.978**	-	-
COIL-full	BERT	0.355	0.963	**0.704**	-
ColBERT	BERT	0.360	0.968	0.694	0.830
CSurF	BERT	0.359	0.970	0.702	**0.849**
CSurF	CoCondenser	**0.374**	**0.978**	**0.704**	0.845
Models with distillation and hard negative mining					
CoCondenser + HN	-	0.382	0.984	0.674	0.820
SPLADE++	CoCondenser	0.380	0.982	0.732	0.875
ColBERT-v2	CoCondenser	**0.397**	0.984	0.744	**0.882**
CSurF$_{HN}$	CoCondenser	0.396	**0.985**	**0.745**	0.880

3.5 Model Implementation and Training

Following previous work, CSurF initializes the LM base with coCondenser [14], which is trained on a retrieval objective. At the encoding stage, we follow the implementation of Formal et al. [11] to generate surface form expansion weights.

$$\phi_v(x) = ReLU(W_v^T x + b_v^T)$$
$$\phi_e(x) = log(1 + ReLU(W_M^T x + b_M^T))$$

The projection layers ϕ_v and ϕ_e are used for both the query and the document. The semantic projection layer W_v, b_v is trained from scratch, while the expansion projection layer W_M, b_M can be initialized from the Masked language modeling (MLM) layer of the LM base.

CSurF is trained end-to-end on the ranking objective with a contrastive loss. Given a query q and a set of n documents $D = \{d^+, d_1^-, ..., d_{n-1}^-\}$, the training loss is

$$\mathcal{L}_{ret} = -log\frac{e^{\mathcal{S}(q,d^+)}}{e^{\mathcal{S}(q,d^+)} + \sum_{i=1}^{n-1} e^{\mathcal{S}(q,d_i^-)}}$$

where $\mathcal{S}$ is the scoring function. CSurF also applies a FLOPS [33] regularization loss to control the sparsity of the expansion weight matrix $\mathbf{E}$, and the number of expanded surface forms. The final loss is the sum of the retrieval loss and the regularization losses of the query and document.

$$\mathcal{L}_{reg}^{\mathbf{q}} = \sum_{t \in V} (w_t^{\mathbf{q}})^2 \quad \mathcal{L}_{reg}^{\mathbf{d}} = \sum_{t \in V} (w_t^{\mathbf{d}})^2$$

$$\mathcal{L} = \mathcal{L}_{ret} + \lambda_q \mathcal{L}_{reg}^{\mathbf{q}} + \lambda_d \mathcal{L}_{reg}^{\mathbf{d}}$$

We first train CSurF on the MSMARCO dataset [29] with training data sampled from a BM25 ranking. Additionally, we boost model performance by incorporating hard negative mining and knowledge distillation [25], where we sample hard negative training triplets from CSurF itself, utilize a cross-encoder re-ranker teacher[1] to generate (q, d) scores, and train a new CSurF model on the sampled training data with an additional KL-divergence loss [36], which aims to minimize the relevance distributions between the cross-encoder teacher and the trained CSurF:

$$\tilde{s}_c(q,d) = \frac{e^{\mathcal{S}_c(q,d)}}{\sum_{d_i \in D} e^{\mathcal{S}_c(q,d_i)}} \quad \tilde{s}(q,d) = \frac{e^{\mathcal{S}(q,d)}}{\sum_{d_i \in D} e^{\mathcal{S}(q,d_i)}}$$

$$\mathcal{L}_{KL} = \sum_{d \in D} \tilde{s}(q,d) log \frac{\tilde{s}(q,d)}{\tilde{s}(q,d)}$$
$$\mathcal{L}_{\text{HN}} = \mathcal{L}_{KL} + \mathcal{L}_{ret} + \lambda_q \mathcal{L}_{reg}^{\mathbf{q}} + \lambda_d \mathcal{L}_{reg}^{\mathbf{d}}$$

where $\mathcal{S}_c$ is the score of the cross-encoder teacher. In the rest of the paper, we use CSurF$_{HN}$ to refer to CSurF models trained with hard negatives and distillation.

4 EXPERIMENTAL METHODOLOGY

Implementation. The retrieval framework of CSurF was built upon the implementation of COIL [2] and with Pytorch [34] and Huggingface [44]. Aside from ablation studies discussed in Section 5.1, we set the semantic representation dimension $|\mathbf{v}| = 32$ with cosine similarity scoring. At indexing time, we prune CSFs with lexical form weight w<1e-8 and organize CSFs in inverted lists in matrix format. Given a query, CSurF (i) performs score calculation for each query CSF and all document CSFs with the same lexical form, (ii) scatters each CSF-CSF match score to the source query-document term match with max reduction, and (iii) performs max-sum score aggregation to calculate the final query-document score.

[1] https://huggingface.co/cross-encoder/ms-marco-MiniLM-L-12-v2
[2] https://github.com/luyug/COIL

Training. We first train CSurF on BM25 negatives on a single GPU for 6 epochs, with 6 queries per batch, 8 documents per training sample (1 positive and 7 negative), and λ_q=λ_d=1e-4. We train CSurF$_{HN}$ variants on 4 GPUs for 8 epochs, with 6 queries per batch and 12 documents per training sample. For simplicity, all CSurF$_{HN}$ variants are trained with the same set of hard negatives sampled from the top 1000 of CSurF rankings on MSMARCO training queries. For all experiments we utilize in-batch negatives for better training. We discuss the effect of tuning λ_q and λ_d in Section 5.2.

Evaluation. We train and evaluate CSurF's in-domain retrieval performance on the MSMARCO passage dataset and on two sets of queries, the MSMARCO Dev query set, and the TREC DL 2019 [5] test queries. Following previous work, we report MRR@10 and Recall@1000 on MSMARCO dev, and NDCG@10 and Recall@1000 for TREC 19 queries. We also perform out-of-domain retrieval experiments on the BEIR benchmark [40], which include multiple datasets with drastically different retrieval settings, domains, and document content. We report model performance on 13 BEIR datasets, with NDCG@10 as the official metric.

Baselines. We mainly compare the performance of CSurF to lexical matching retrieval systems, including: (1) BM25 [37] and BM25 with DocT5Query augmentation [31] (2) lexical exact-match systems COIL-tok [15] and SPLADE [11, 12] (3) lexical all-to-all soft-match system ColBERT [21, 39]. Note that in this work we do not perform extended model and training setup design as discussed in recent works to improve learned sparse retrieval training [22, 28], and compare CSurF's performance to the original results reported for SPLADE++ and ColBERT-v2, which uses the same or comparable training setups. We also include the performance of two dense retrieval systems [14, 19] and the hybrid COIL system which performs hybrid retrieval with dense and lexical components. We separately evaluate and compare systems trained without and with knowledge distillation and hard negative mining.

5 EXPERIMENTAL RESULTS

In this section, we report CSurF's in-domain retrieval performance on MSMARCO in Sections 5.1 and 5.2, respectively focusing on retrieval effectiveness and efficiency. Section 5.3 discusses out-of-domain retrieval performance. Section 5.4 concludes the experiments with detailed case study of generated CSFs.

5.1 In-domain passage retrieval effectiveness

Table 1 reports CSurF's retrieval performance on MSMARCO. Without knowledge distillation and under comparable LM settings, CSurF outperforms current lexical exact match systems, including SPLADE and COIL-tok, on both recall and accuracy (MRR@10). With hard-negative mining and distillation, CSurF still outperforms SPLADE++ on MRR@10. This demonstrates CSurF's ability to bridge the vocabulary and semantic mismatch in lexical exact match. Furthermore, the sparse retrieval framework CSurF consistently reaches the performance level of the all-to-all soft-match ColBERT in all training settings, with much lower retrieval complexity. We analyze the retrieval cost and effectiveness-efficiency tradeoff for CSurF in Section 5.2.

Table 2: Ablation study for CSurF and comparison to current lexical exact-match systems. All models initialized with co-Condenser. CSurF models are trained with λ_q=λ_d=1e-3. The "Full" model capacity indicates using CSFs from both original text and expansion ($\mathcal{O} + \mathcal{E}$), and vector scoring with $|\mathbf{v}| = 32$.

Model		MSMARCO Dev	
Retriever	Variant	MRR@10	R@1000
CSurF	Full	0.374	0.978
CSurF$_{HN}$	Full	0.395	0.987
COIL-tok		*0.353*	*0.949*
CSurF	$\mathcal{O}$ only	0.359	0.955
CSurF	$\mathcal{E}$ only	0.369	0.979
CSurF$_{HN}$	$\mathcal{O}$ only	0.374	0.969
CSurF$_{HN}$	$\mathcal{E}$ only	0.391	0.987
SPLADE++		*0.380*	*0.982*
CSurF$_{HN}$	$\|\mathbf{v}\|$=32, dot	0.395	0.987
CSurF$_{HN}$	$\|\mathbf{v}\|$=32, cos	0.396	0.985
CSurF$_{HN}$	$\|\mathbf{v}\|$=4, cos	0.395	0.985
CSurF$_{HN}$	$\|\mathbf{v}\|$=0 (w only)	0.391	0.985

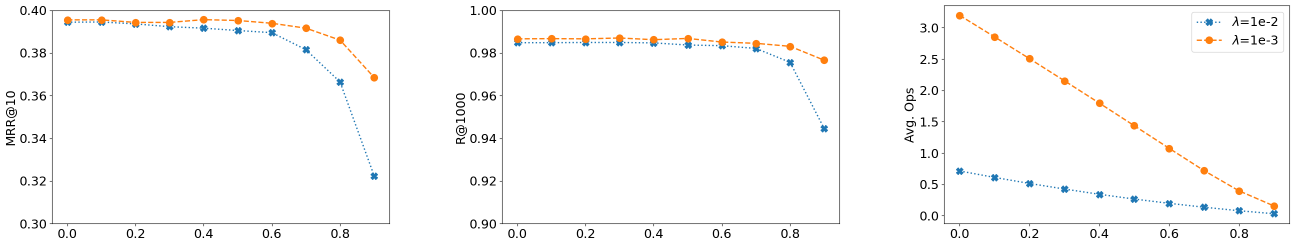
We further perform two sets of ablation studies on model components of CSurF, to investigate the connection and comparison between CSurF and existing lexical exact-match systems. Namely, we look into the effect of two aspects discussed in Section 3.4: lexical form expansion and contextual semantics grounding.

Table 2 reports the retrieval performance on MSMARCO for CSurF variants. To look into the effect of lexical expansion, we train CSurF models which only generate CSFs from original-text-based lexical forms $\mathcal{O}$ or only from expansion-based lexical forms $\mathcal{E}$. Compared to solely performing exact-match over the original text, introducing lexical form expansion improves model performance by a wide margin. Specifically, it is the primary source of gain in recall (0.955 to 0.975 without distillation). Without the expansion component, the model capacity of CSurF is limited, even with extra knowledge distillation from the cross-encoder teacher. This suggests that lexical exact-match signals of original text terms are naturally not sufficient to accurately predict query-document relevance, and it is critical to introduce extra matching signals.

To look into the effect of contextualized representations and term scoring, we experiment with four model variants, which include: (i) three vector representation settings where $|\mathbf{v}| = 32$ with the term scoring function $\mathbf{f}$ as dot product and cosine similarity, and $|\mathbf{v}| = 4$ with cosine scoring, and (ii) a term weighting-based model variant where $|\mathbf{v}| = 0$ (i.e. only the lexical expansion weight w is used). We see from Table 2 that all systems perform similarly on recall, with multi-vector systems outperforming term-weighting systems in accuracy (MRR@10). This result together with the shown result in Table 1 demonstrates the benefit of expanding sparse retrieval systems such as SPLADE to multi-vector systems. We further note that the lexical weight-only CSurF variant still outperforms the current SPLADE++ model in accuracy, and is slightly better than the highest reported results in recent SPLADE-related works [22],

Table 3: Model efficiency analysis. Numbers labeled with * indicate the original MARCO dev query/document lengths after BERT tokenization. Other lengths stats are calculated after pruning inverted index entries with 0 term weight.

Model		MSMARCO Dev		Efficiency		
Retriever	Variant	MRR@10	R@1000	Avg. Q Len	Avg. D Len	Avg Ops
COIL-tok		0.353	0.949	*6.9**	*63.4**	2.28
ColBERT-v2		0.397	0.984	*6.9**	*63.4**	-
CSurF$_{HN}$	λ_q=1e-3, λ_d=1e-3	0.396	0.985	49.6	230	3.20
CSurF$_{HN}$	λ_q=1e-2, λ_d=1e-2	0.394	0.984	23.9	98	0.71
CSurF$_{HN}$	λ_q=1e-1, λ_d=1e-1	0.388	0.983	11.6	37	0.16
CSurF$_{HN}$	λ_q=5e-1, λ_d=5e-1	0.377	0.977	9.9	22	0.07
CSurF$_{HN}$	λ_q=1e-2, λ_d=1e-3	0.395	0.986	21.2	207	1.70
CSurF$_{HN}$	λ_q=1e-1, λ_d=1e-3	0.397	0.984	11.2	173	1.02
CSurF$_{HN}$	λ_q=5e-1, λ_d=1e-3	0.396	0.984	10.6	174	0.96



(a) MRR@10 **(b) Recall@1000** **(c) Average operations**

Figure 2: Performance of CSurF with different pruning thresholds.

which performs in-depth analysis of applying different model training techniques, such as regularization and separate query encoder, to improve SPLADE performance. This demonstrates the effectiveness of the additional grounding step from lexical form to context source in CSurF, which allows the match performance of original terms to directly guide the learning of expansion terms at training time. This also shows that CSurF's performance could be potentially further improved with the application of training techniques discussed above.

5.2 Retrieval Efficiency

This section discusses the effectiveness-efficiency trade-off in the retrieval procedure of CSurF. As a sparse retrieval system, CSurF represents a query or document with a set of CSFs. Without changing the dimensionality of the semantic representations, the indexing storage cost of CSurF is directly determined by the number of CSFs generated per document, and the retrieval computational cost is further determined by the number of CSF matches given a (q, d) pair. Therefore, we report the following metrics to compare the run cost of CSurF: (i) the average number of terms to represent the query and document, i.e. $|C^q|$ and $|C^d|$, and (ii) the average number of "term matches", or term-scoring operations required given a (q, d) pair [12]. For exact-match systems such as CSurF, this is estimated by $\mathbb{E}_{OPS} = \sum_{t \in V}(\hat{n}_t^q \hat{n}_t^d)$ where $\hat{n}_t^q$ and $\hat{n}_t^d$ denote the average number of occurrences for token t in a query or document.

At training time, the number of generated CSFs is mainly affected by the regularization weights λ_q and λ_d which control the FLOPS penalty and thus the sparsity of the expansion weights. We list the performance of CSurF with different λ_q and λ_d settings in Table 3. CSurF is able to maintain high model capacity with a relatively small bag-of-CSFs size, achieving >0.39 MRR@10 with (3.5×, 1.5×) or (1.5×, 2.7×) the lengths of the original query and document. More importantly, CSurF learns sparse matching signals without degradation in model capacity. Compared to current multi-vector retrievers, CSurF significantly outperforms the exact-match retriever COIL-tok, while requiring comparable and often fewer retrieval time matching operations. On the other hand, CSurF reaches comparable model effectiveness to ColBERT-v2, with a significantly lower calculation operation count than all-to-all soft match of all tokens (>400 ops.), and not requiring further filtering stages of candidate passage selection as introduced in ColBERT-v2.

We also experiment with post-hoc CSF inverted index pruning, where we explicitly prune $\alpha \in [0, 1)$ of the encoded corpus, removing document CSFs with the lowest lexical form expansion weights. Figure 2 reports the performance change of two CSurF models trained with λ_q=λ_d=1e-2 and λ_q=λ_d=1e-3 with cosine similarity scoring, with different pruning threshold α. We see that pruning further reduces the number of redundant CSFs and match signals for both models, leading to further retrieval efficiency improvement while maintaining retrieval capacity (MRR@10>0.39, R@1000>0.98 with 50% of terms pruned).

Table 4: Retreival performance (NDCG@10) on 13 BEIR datasets. Best zero-shot performance on each dataset is labeled in bold. CSurF$_{HN}$ (γ=0.0) denotes the original CSurF performance without interpolation. CSurF$_{HN}$-oracle performance is underlined when equal to or better than the best baseline.

Model	AA	CF	DB	FE	FQ	HQ	NF	NQ	QU	SD	SF	T2	TC	Avg.
BM25	0.441	0.179	0.288	0.648	0.239	0.601	0.297	0.310	-	0.156	0.620	-	0.616	-
Contriever	0.446	**0.237**	0.413	0.758	0.329	0.638	0.328	0.498	-	**0.165**	0.677	0.230	0.596	-
ColBERTv2	0.463	0.176	0.446	0.785	0.356	0.667	0.338	**0.562**	0.854	0.154	0.693	**0.263**	**0.738**	0.500
SPLADE++	0.518	**0.237**	0.436	**0.796**	0.349	**0.693**	0.345	0.533	0.849	0.161	**0.710**	0.242	0.725	0.507
CSurF$_{HN}$ (γ = 0.0)	**0.521**	0.186	**0.453**	0.720	**0.361**	**0.693**	**0.351**	0.543	**0.861**	0.158	0.708	0.262	0.726	0.503
CSurF$_{HN}$ (γ = 1.0)	0.352	0.159	0.420	0.692	0.329	0.690	0.310	0.505	0.833	0.154	0.690	0.232	0.696	0.466
CSurF$_{HN}$ (oracle)	*0.521*	*0.192*	*0.458*	*0.729*	*0.370*	*0.713*	*0.353*	*0.550*	*0.867*	*0.161*	*0.717*	*0.264*	*0.739*	*0.510*
(Best γ)	*0.0*	*0.3*	*0.1*	*0.2*	*0.2*	*0.4*	*0.1*	*0.2*	*0.2*	*0.3*	*0.2*	*0.2*	*0.2*	

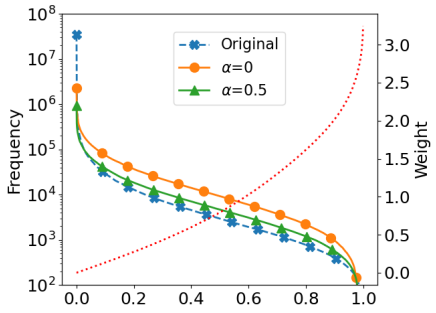


Figure 3: Lexical form frequency and weights for CSurF. The red dotted curve denotes weight distribution. Model trained with λ_q=λ_d=1e-2. X axis denotes percentage of corpus CSFs.

These experiments demonstrate that the CSurF retrieval process can be highly efficient due to the sparsity of CSF match signals. To look into the properties of the CSF generation and matching processes, we analyze and plot the distribution of corpus CSFs' lexical form frequency and expansion weight in Figure 3. Compared to the lexical form frequency distribution of the original text, CSurF is trained to simultaneously expand meaningful lexical surface forms but also prune existing lexical terms with low term importance, and removes a significant proportion of tokens with the highest occurrence frequency, most of which do not carry important contextual meaning such as stop words. This leads to the aforementioned comparison where CSurF requires lower scoring operations per query than COIL-tok despite having "longer" queries or documents. Post-hoc index pruning with $\alpha = 0.5$ further removes redundant matching signals of lexical forms at all frequencies, resulting in a significant decrease of matching operations without major influence in retrieval performance. For instance, after training and post-hoc pruning, the five most frequent terms in the MSMARCO passage set ("the", "of", "and", "in", "to") are removed by over 98.5% compared to their original corpus term frequency.

Finally, we note that CSurF utilizes vector term representations, and the storage and run cost of CSurF is affected by the representation dimension $|\mathbf{v}|$ and the computational cost of the actual term-scoring operation $\mathbf{f}()$. As listed in Table 2, we recognize the tradeoff in performance where a higher representation dimension leads to improved performance. Many recent works have also targeted improving the efficiency of representation storage and scoring in multi-vector retrieval, with proposed methods such as semantic representation clustering and residual compression [10, 38, 39] also compatible with CSurF. We leave detailed engineering and optimization of the vector scoring step as future work.

5.3 Out-of-domain retrieval

In this section, we evaluate CSurF on out-of-domain retrieval. Table 4 lists the retrieval performance of CSurF on 13 datasets in the BEIR benchmark. Compared to baseline approaches, CSurF achieves best performance on 6 of 13 datasets, with a win-loss-tie of 10:3:0 and 8:4:1 compared to ColBERT-v2 and SPLADE++ respectively. We also observed very different trends in performance across different datasets. Specifically, CSurF has low performance on Climate-Fever [9] and Fever [41]. This may be related to the property of the retrieval tasks and queries focusing more on exact match of specific entities, and vector representations introducing noise.

Based on previous work which discusses the effect of original lexical terms in zero-shot retrieval settings, we utilize CSurF's capability to track lexical form source (original text or expansion) and perform an extra experiment to explicitly introduce and emphasize the original text lexical form information. We experiment with a weight-based interpolation approach for CSurF scoring, where at retrieval time, we modify the lexical form weights of CSFs and apply a penalty to all CSFs generated solely via expansion, i.e. $w_A^* = (1-\gamma)w_A + \gamma\mathbb{I}_{(A\in\mathbf{O})}w_A$, where $\gamma \in [0,1]$ is the penalty parameter. $\gamma = 0$ represents the original CSurF performance without interpolation, while a higher γ indicates lower confidence or larger penalty on expansion-based CSFs.

We test γ=[0.0,0.1,...,0.9,1.0] and list the *oracle* performance of CSurF after interpolation and the corresponding γ in Table 4. On most datasets, the oracle performance of CSurF is achieved at γ=0.1-0.3. This demonstrates a mixed message: CSurF is overall effective in expanding meaningful surface forms even in zero-shot settings, but the LM backbone and expansion component may still suffer from the change of retrieval domain, and interpolation with an original-text-only retrieval source or explicitly emphasizing original text importance can still be helpful.

Table 5: Examples of bag-of-CSFs. We show a query and document from MSMARCO and a query from FEVER in this order. Original terms are marked in bold. The lexical form and lexical weights of generated CSFs are listed after its source term. CSFs removed by post-hoc pruning ($\alpha = 0.5$) are presented in gray, and remaining CSFs are underlined.

how **long** (time 2.13) (length 1.64) (duration 1.36) (long 1.19) (hours 0.63) (longest 0.62) (distance 0.47) (times 0.1) (minutes 0.09) **is** (is 0.45) **super** (super 2.05) **bowl** (bowl 2.23) (bowls 1.01) (nfl 0.38) (final 0.01) **game** (game 1.45)
jefferson (jefferson 2.71) (napoleon 1.13) (adams 0.54) (louis 0.51) (lafayette 0.45) (lewis 0.32) (clark 0.09) **should** (should 1.31) (deserved 0.37) (deserve 0.01) **right** (right 1.35) **##ly** **be** (be 0.31) (best 0.18) **remembered** (remembered 2.23) (remember 1.49) (honored 0.93) (forgotten 0.85) (remembering 0.79) (recognized 0.72) (recalled 0.15) (memory 0.14) (celebrated 0.13) (acknowledged 0.07) **for** (for 0.83) (because 0.36) **the** **great** (great 1.54) **deed** (deed 1.74) (deeds 0.36) **of** **purchasing** (purchasing 1.56) (buying 0.94) (bought 0.76) (purchased 0.56) (acquiring 0.42) (acquisition 0.22) **this** (monroe 1.04) (texas 0.34) (france 0.18) (davis 0.14) (west 0.05) (this 0.04) **enormous** (enormous 1.47) (massive 0.66) (immense 0.52) (vast 0.27) (expansive 0.19) (extensive 0.18) **tract** (tract 1.15) **of** (expansion 0.34) (conquer 0.2) (of 0.0) **land** (land 1.68) .....
death (death 2.89) (died 1.61) (murder 1.2) (deaths 1.17) (suicide 1.13) (fatal 1.04) (dead 0.83) (tragedy 0.65) (die 0.15) **note** (note 2.87) (notes 2.04) (##note 1.28) (wo 0.41) **is** (is 0.43) (manga 0.26) **a** (genre 0.4) **japanese** (japanese 2.04) (japan 1.82) (anime 1.39) (late 0.14) (tokyo 0.14) **television** (television 1.24) (tv 1.17) **drama** (drama 2.0) (comedy 0.11) **series** (series 1.62) (show 0.21) **that** (movie 0.5) **first** (first 0.74) (debut 0.58) (premiere 0.28) **aired** (aired 1.8) (air 0.87) (premiered 0.65) (airing 0.07) **in** (date 1.51)(episode 0.92) (season 0.84) **2015** (2015 1.91) (singapore 0.33)

5.4 Case study

In this section, we present 3 detailed examples of the generated bag-of-CSFs in Table 5. We select a query and document from the MSMARCO passage dataset, and a query from the FEVER dataset. We observe that CSurF possesses the ability to understand the context and expand surface forms that are related to the original term (e.g. plural forms or acronyms), express the same concept ("duration" and "length" expanded from "how long"), or other terms in related fields ("nfl" and "final" expanded from "super bowl", which are all related to concepts in American football). It also assigns higher weights to important CSFs such as core entities, and assigns lower weights or directly prunes terms like "how", "is" and "the", matching the analysis in Section 5.2.

We also discuss two interesting observations which points to directions of further improvement of CSurF. For proper nouns such as person names, dates or locations, CSurF may not understand the entity name (e.g. "Death Note" as a TV series), or may over-generate lexical terms which refer to the same type of entities but irrelevant to the current context (e.g. generating names from original term "Jefferson"). These are classic problems in exact-match-based systems, and potential solutions include injecting external knowledge to correctly distinguish and generate related entities, and rethinking and refining the post-hoc pruning stage for CSF-selection. We also observe that CSurF occasionally generates relevant lexical surface forms from the seemingly "incorrect" source such as stop words. This does not effect surface form matching but may affect the accuracy of vector term representation and scoring. In the FEVER example CSurF learns to prune the original term "a" and "in", but still generates surface forms "genre" and "date" from such terms. This calls for a deeper analysis of the lexical expansion step, but also raises a potential model extension, where we introduce extra sources to generate lexical forms related to the *overall concept or topic* of the text sequence with an independent representation.

6 CONCLUSION

This paper proposes CSurF, which performs sparse lexicon-based retrieval through constructing and matching Contextualized Surface Forms. Its retrieval process combines efficient surface form exact match and fine-grained contextualized semantic scoring, which leads to maximized model capacity while maintaining the simplicity and efficiency of exact-match-based retrieval systems.

CSurF extends current term-weight based learned sparse retrieval approaches with vector term representations. On experiments across multiple datasets and retrieval settings, CSurF is able to simultaneously bridge the vocabulary and semantic mismatch in exact-match retrieval, and achieve state-of-the-art retrieval performance for lexical exact-match systems. Ablation studies and analysis further demonstrate CSurF's ability to jointly expand meaningful surface forms and ground surface forms to underlying semantics, which leads to increased model capacity. We also propose a simple interpolation approach in out-of-domain retrieval settings, to analyze the effect of original text vs. expanded surface forms as well as the quality of lexical form expansion on different retrieval tasks.

Compared to all-to-all soft-match retrievers, CSurF achieves comparable performance across all retrieval tasks as an exact-match-based retrieval system. CSurF is able to learn sparse connections of the original query and document terms, resolving the key efficiency issue of lexical soft-match. The retrieval efficiency of CSurF can also be further optimized with different approaches including training regularization adjustment, post-hoc index pruning, and vector representation approximation or dimension control, without significantly affecting retrieval accuracy. We hope this work encourages more research on building effective, efficient, robust and knowledge-enhanced sparse retrieval systems in the real world, as well as exploring the connection and distinction among current retrieval frameworks and systems.

REFERENCES

[1] Nasreen Abdul-Jaleel, James Allan, W Bruce Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Mark D Smucker, and Courtney Wade. 2004. UMass at TREC 2004: Novelty and HARD. *Computer Science Department Faculty Publication Series* (2004), 189.

[2] Yang Bai, Xiaoguang Li, Gang Wang, Chaoliang Zhang, Lifeng Shang, Jun Xu, Zhaowei Wang, Fangshan Wang, and Qun Liu. 2020. SparTerm: Learning term-based sparse representation for fast text retrieval. *arXiv preprint arXiv:2010.00768* (2020).

[3] Bodo Billerbeck and Justin Zobel. 2005. Document expansion versus query expansion for ad-hoc retrieval. In *Proceedings of the 10th Australasian Document Computing Symposium*. 34–41.

[4] Guihong Cao, Jian-Yun Nie, Jianfeng Gao, and Stephen Robertson. 2008. Selecting good expansion terms for pseudo-relevance feedback. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. 243–250.

[5] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M Voorhees. 2020. Overview of the TREC 2019 deep learning track. *arXiv preprint arXiv:2003.07820* (2020).

[6] Zhuyun Dai and Jamie Callan. 2020. Context-aware document term weighting for ad-hoc search. In *Proceedings of The Web Conference 2020*. 1897–1907.

[7] Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. 2018. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *Proceedings of the eleventh ACM international conference on web search and data mining*. 126–134.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[9] Thomas Diggelmann, Jordan Boyd-Graber, Jannis Bulian, Massimiliano Ciaramita, and Markus Leippold. 2020. Climate-fever: A dataset for verification of real-world climate claims. *arXiv preprint arXiv:2012.00614* (2020).

[10] Zhen Fan, Luyu Gao, Rohan Jha, and Jamie Callan. 2023. COILcr: Efficient Semantic Matching in Contextualized Exact Match Retrieval. In *Advances in Information Retrieval: 45th European Conference on Information Retrieval, ECIR 2023, Dublin, Ireland, April 2–6, 2023, Proceedings, Part I*. Springer, 298–312.

[11] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE v2: Sparse lexical and expansion model for information retrieval. *arXiv preprint arXiv:2109.10086* (2021).

[12] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse lexical and expansion model for first stage ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2288–2292.

[13] Luyu Gao and Jamie Callan. 2021. Condenser: a pre-training architecture for dense retrieval. *arXiv preprint arXiv:2104.08253* (2021).

[14] Luyu Gao and Jamie Callan. 2021. Unsupervised corpus aware language model pre-training for dense passage retrieval. *arXiv preprint arXiv:2108.05540* (2021).

[15] Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. COIL: Revisit Exact Lexical Match in Information Retrieval with Contextualized Inverted List. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*. Association for Computational Linguistics, 3030–3042. https://doi.org/10.18653/v1/2021.naacl-main.241

[16] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821* (2021).

[17] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. Semantic matching by non-linear word transportation for information retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 701–710.

[18] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 2021. Efficiently teaching an effective dense retriever with balanced topic aware sampling. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 113–122.

[19] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Towards unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118* (2021).

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

[21] Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 39–48.

[22] Carlos Lassance and Stéphane Clinchant. 2022. An efficiency study for SPLADE models. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2220–2226.

[23] Minghan Li, Sheng-Chieh Lin, Barlas Oguz, Asish Ghoshal, Jimmy Lin, Yashar Mehdad, Wen-tau Yih, and Xilun Chen. 2022. CITADEL: Conditional Token Interaction via Dynamic Lexical Routing for Efficient and Effective Multi-Vector Retrieval. *arXiv preprint arXiv:2211.10411* (2022).

[24] Jimmy Lin and Xueguang Ma. 2021. A Few Brief Notes on DeepImpact, COIL, and a Conceptual Framework for Information Retrieval Techniques. *CoRR* abs/2106.14807 (2021). arXiv:2106.14807 https://arxiv.org/abs/2106.14807

[25] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2021. In-batch negatives for knowledge distillation with tightly-coupled teachers for dense retrieval. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*. 163–173.

[26] Donald Metzler and W Bruce Croft. 2005. A markov random field model for term dependencies. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*. 472–479.

[27] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems* 26 (2013).

[28] Thong Nguyen, Sean MacAvaney, and Andrew Yates. 2023. A Unified Framework for Learned Sparse Retrieval. In *Advances in Information Retrieval: 45th European Conference on Information Retrieval, ECIR 2023, Dublin, Ireland, April 2–6, 2023, Proceedings, Part III*. Springer, 101–116.

[29] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *CoCo@ NIPS*.

[30] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *arXiv preprint arXiv:1901.04085* (2019).

[31] Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. 2019. From doc2query to docTTTTTquery. *Online preprint* 6 (2019).

[32] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document expansion by query prediction. *arXiv preprint arXiv:1904.08375* (2019).

[33] Biswajit Paria, Chih-Kuan Yeh, Ian EH Yen, Ning Xu, Pradeep Ravikumar, and Barnabás Póczos. 2020. Minimizing flops to learn efficient sparse representations. *arXiv preprint arXiv:2004.05665* (2020).

[34] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[35] Yujie Qian, Jinhyuk Lee, Sai Meher Karthik Duddu, Zhuyun Dai, Siddhartha Brahma, Iftekhar Naim, Tao Lei, and Vincent Y Zhao. 2022. Multi-Vector Retrieval as Sparse Alignment. *arXiv preprint arXiv:2211.01267* (2022).

[36] Ruiyang Ren, Yingqi Qu, Jing Liu, Wayne Xin Zhao, Qiaoqiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. 2021. Rocketqav2: A joint training method for dense passage retrieval and passage re-ranking. *arXiv preprint arXiv:2110.07367* (2021).

[37] Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* 3, 4 (2009), 333–389.

[38] Keshav Santhanam, Omar Khattab, Christopher Potts, and Matei Zaharia. 2022. PLAID: an efficient engine for late interaction retrieval. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 1747–1756.

[39] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2021. Colbertv2: Effective and efficient retrieval via lightweight late interaction. *arXiv preprint arXiv:2112.01488* (2021).

[40] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663* (2021).

[41] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a Large-scale Dataset for Fact Extraction and VERification. In *NAACL-HLT*.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[43] Ellen M Voorhees. 1994. Query expansion using lexical-semantic relations. In *SIGIR'94: Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, organised by Dublin City University*. Springer, 61–69.

[44] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771* (2019).

[45] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR conference on research and development in information retrieval*. 55–64.

[46] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808* (2020).

[47] Jinxi Xu and W Bruce Croft. 1996. Query expansion using local and global document analysis. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*. 4–11.

[48] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Optimizing dense retrieval model training with hard negatives. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1503–1512.

[49] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2022. Learning Discrete Representations via Constrained Clustering for Effective and Efficient Dense Retrieval. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining* (Virtual Event, AZ, USA) *(WSDM '22)*. Association for Computing Machinery, New York, NY, USA, 1328–1336. https://doi.org/10.1145/3488560.3498443

[50] Tiancheng Zhao, Xiaopeng Lu, and Kyusong Lee. 2021. SPARTA: Efficient Open-Domain Question Answering via Sparse Transformer Matching Retrieval. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Online, 565–575. https://doi.org/10.18653/v1/2021.naacl-main.47