Learning to Re-Rank with Contextualized Stopwords

Sebastian Hofstätter
TU Wien
s.hofstaetter@tuwien.ac.at

Aldo Lipani
University College London
aldo.lipani@ucl.ac.uk

Markus Zlabinger
TU Wien
markus.zlabinger@tuwien.ac.at

Allan Hanbury
TU Wien
hanbury@ifs.tuwien.ac.at

ABSTRACT

The use of stopwords has been thoroughly studied in traditional Information Retrieval systems, but remains unexplored in the context of neural models. Neural re-ranking models take the full text of both the query and document into account. Naturally, removing tokens that do not carry relevance information provides us with an opportunity to improve the effectiveness by reducing noise and lower document representation caching-storage requirements. In this work we propose a novel contextualized stopword detection mechanism for neural re-ranking models. This mechanism consists of training a sparse vector in order to filter out document tokens from the ranking decision. This vector is learned end-to-end based on the contextualized document representations, allowing the model to filter terms on a per occurrence basis. This leads to a more explainable model, as it reduces noise. We integrate our component into the state-of-the-art interaction-based TK neural re-ranking model. Our experiments on the MS MARCO passage collection and queries from the TREC 2019 Deep Learning Track show that filtering out traditional stopwords prior to the neural model reduces its effectiveness, while learning to filter out contextualized representations improves it.

1 INTRODUCTION

Filtering stopwords is an established technique in Information Retrieval (IR). In many domains stopwords reduce noise and increase the efficiency of an inverted index [6, 21]. In this paper we examine the use of stopwords, namely filtering out terms from the scoring decision, in neural re-ranking models. In particular, we focus on interaction-based re-ranking models [7], where each query term interacts with each document term, since they offer an opportunity to modify individual interactions between document and query terms. Recent advances in neural re-ranking using Transformers, like BERT [16] and TK [9], showed strong effectiveness gains, thanks to their ability to contextualize the input sequence. To offset the high computational cost of contextualized document representations, some models, such as the TK model, allow to pre-compute and store document vectors. The storage requirement grows linearly with the total number of terms, saving less terms means a smaller storage.

We propose a novel sparse-stopword component for interaction-based neural re-ranking models, which dynamically and explicitly removes document tokens from the decision layer based on their contextualized representation. We integrate our component into the TK [9] model, resulting in a new model which we call *TK-Sparse*.

Term importance and stopwords have a long history in IR: from the collection occurrence based Inverse Document Frequency (IDF) [17], the theory of retrievability [1, 13], to studies on the impact of stopwords in retrieval and text processing [6, 21], and visual inspection tools of neural re-ranking models [10]. Term importance in neural models has been studied in the context of query attention [7, 19] and IDF weighting [11], or BERT-based importance measures for indexing [4]. However, we are the first to study the explicit filtering of contextualized terms from the ranking decision.

We train the sparse-stopword component by augmenting the margin ranking loss with an L1-norm of the sparsity vector, including a weighting factor to control the sparsity. The successful optimization requires adaptations to the training process, which we also discuss to help establish sparsity techniques in the neural-IR community. Using the established technique of minimizing the L1-norm to increase a machine learning models sparsity is gaining popularity among neural networks. In the context of IR, Zamani et al. [22] utilized sparse representations for indexing, in contrast to our work their goal was to create single document vector.

We conduct experiments on the MS MARCO passage collection with two distinct query sets (many queries with limited and fewer queries with dense judgments from the TREC 2019 Deep Learning track). We show that removing traditional, Lucene and collection frequency based, stopwords reduces the effectiveness of the TK model. However, its augmented version, the TK-Sparse model, with contextualized stopwords, significantly increases the effectiveness with up to 40 % of words removed. When pre-computing document representations, these removed stopwords transfer directly to 40 % storage savings for contextualized document representations.

Finally, we analyze the distributions of removed contextualized terms in relation to traditional stopwords. We observe an overlap, however TK-Sparse removes traditional terms less often and also removes additional words not part of traditional stopwords.

In summary, the main contributions of this work are:

- We present a novel contextualized stopword component, and instructions to train it with the TK neural ranking model;
- We demonstrate the effectiveness gains of contextualized stopwords with an ablation of different sparsity levels;
- We provide a thorough analysis of the contextualized stopword occurrence patterns;
- We publish our documented implementation at: *github.com/sebastian-hofstaetter/transformer-kernel-ranking*

2 CONTEXTUALIZED STOPWORDS

In this section, we present our contextualized stopword component as an adaptation of the TK model (Section 2.1) and required

adaptations to the training process to successfully train the sparsity component in the context of IR (Section 2.2).

2.1 TK-Sparse: Neural Ranking with Sparsity

Our novel sparse-stopword component can be integrated in every interaction-based model. We integrated our component into the interaction model TK [9], and refer to it as TK-Sparse.

As the first step, query ($q_{1:n}$) and document embeddings ($d_{1:m}$) are independently contextualized with a shallow Transformer (TF):

$$\begin{aligned} \hat{q}_{1:n} &= \text{TF}(q_{1:n}) \\ \hat{d}_{1:m} &= \text{TF}(d_{1:m}) \end{aligned} \tag{1}$$

The independence of the contextualization allows us to utilize the contextualized document representations for our stopword component. Every vector $\hat{d}_j$ is transformed by two linear layers (with weight matrices W_1, W_2 and bias vectors b_1, b_2), followed by a ReLU activation, to compute the stopword removal gate r_j:

$$r_j = \text{ReLU}(\tanh(\hat{d}_j W_1 + b_1)W_2 + b_2) \tag{2}$$

The ReLU sets every negative value to 0, allowing us to cancel out interactions for this document term. As a positive side-effect of the stopword removal we also receive salience information in form of the positive values for every remaining term. The TK model applies kernel-activation (KA) [18] with multiple kernels k to the interaction matrix between every query and document term:

$$K^k_{i,j} = \text{KA}^k(\cos(\hat{q}_i, \hat{d}_j)) \tag{3}$$

We integrate the result of the stopword component after the kernel-activation, as otherwise the zeroed document terms would be counted in the respective kernel. However, we want to completely cancel the kernel-activation for a term to be able to later remove the corresponding vector from the document cache. The integration of the stopword removal gate r_j is an element-wise multiplication with each activation before summing the document activations:

$$K^k_i = \sum_{j=1}^{m} K^k_{i,j} * r_j \tag{4}$$

In the last step of the model we add a scaling value α^k to counteract the reduced value size expectation per kernel required for the log-activation, which is applied to every kernel result before weighting and summing the kernel scores to form the final score:

$$s = \sum_{j=i}^{n} \left(\log \left(K^k_i * \alpha^k \right) \right) W \tag{5}$$

Finally, to actually force the model to learn a sparse removal-vector r we augment the margin ranking loss of a negative document d_{neg} and positive document d_{pos} per query with the L1-norm of the positive & negative r, and a hyperparameter λ to control the importance of the L1 regularization:

$$L(d_{pos}, d_{neg}) = (d_{neg} - d_{pos} + 1) + \lambda * \left(||r^{pos}||_1 + ||r^{neg}||_1 \right) \tag{6}$$

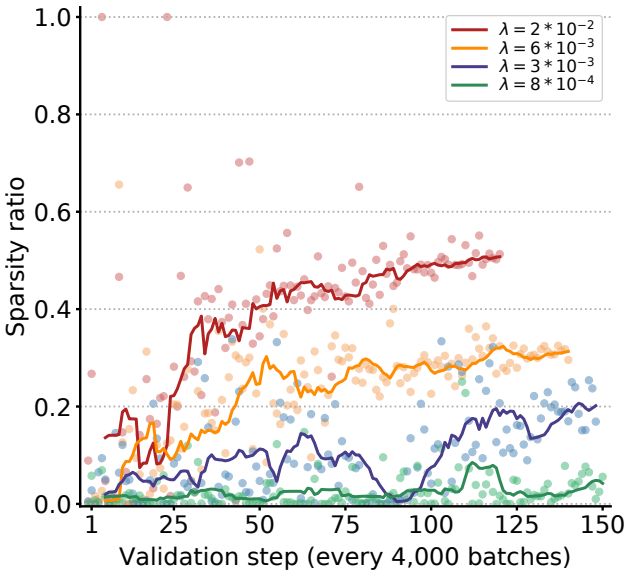

Figure 1: Sparsity ratio of removed tokens in TK-Sparse's validation steps for different λ. The lines represent a 10-point moving average with different lengths due to early stopping.

2.2 Adaptations for Sparse Training

We found the training process of sparse vectors to be cumbersome and we believe it presents a high entry barrier for the community to broadly adopt sparsity as a tool in neural re-ranking models. Therefore, we now discuss our main *tricks* that we employed to successfully train our sparse component:

Initialization We initialize the biases (b_1, b_2) of the stopword component with sufficiently large positive values to prevent the model from starting in a broken and un-recoverable state.

Continuous monitoring We continuously save the average sparsity ratio (i.e. the number of stopwords relative to the total word count) of the training samples, to quickly detect if there are problems with the training. Especially the value of the hyperparameter λ, which if set too low has no effect or if set too high collapses the training quickly by removing all document terms, resulting in a broken model. In Figure 1, we show that at the beginning of the training the sparsity ratio is much more volatile than in later stages, especially after we gradually reduced the learning rate.

Reanimation With the continuous monitoring, we probe the average of the last 100 training batches to detect if the model has collapsed to a sparsity ratio of 100%. If this is the case the model cannot update its state, as no useful gradient can be computed. To overcome this problem we reanimate the stopword component by adding a fixed small positive number to the last bias before the ReLU activation (b_2). We do this as long as the sparsity ratio remains at 100%. Usually, the model quickly recovers and resumes training. This is a quick and efficient way to let the training continue.

Gradient clipping To stabilize the sparsity ratio during training and reduce the number of reanimations we clip the gradients of the linear layers to small values, to prevent sudden changes in the two layers [14]. All our configuration settings are available in our repository.

Table 1: MSMARCO-Passage test set results. *Stop* refers to the percentage of removed words in the tested documents. Significant improvements are indicated with the characters assigned to each TK variant. In bold is the top TK variant per metric.

Sig.	Model	Stop	DEV (limited judgments)			TREC-2019 (dense judgments)				
			nDCG@10	MRR@10	R@10	nDCG@3	nDCG@10	MRR@10	R@10	MAP@1K
	BM25	-	0.241	0.194	0.402	0.521	0.501	0.689	0.185	0.294
	DUET	-	0.299	0.248	0.468	0.598	0.544	0.782	0.191	0.313
	CO-PACRR	-	0.328	0.273	0.514	0.620	0.612	0.755	0.223	0.365
	CONV-KNRM	-	0.332	0.277	0.519	0.627	0.609	0.716	0.223	0.360
a	***TK***	-	^{bcdj}0.369	^{bcdj}0.311	^{bcdj}0.564	0.655	0.649	0.821	**0.242**	0.396
b	***TK /w LuceneStop***	24 %	^{cdj}0.359	^{cdj}0.301	^{cdj}0.555	0.661	0.630	^{dg}0.795	0.220	0.392
c	***TK /w CollectionTop25***	35 %	^{cdj}0.353	^{dj}0.296	^{j}0.547	0.649	0.624	0.756	0.214	^{d}0.388
d	***TK /w CollectionTop50***	41 %	0.350	0.293	0.543	0.635	0.627	0.745	0.230	0.376
e	***TK-Sparse*** $\lambda = 8 * 10^{-4}$	3 %	^{abcdghij}**0.373**	^{abcdhij}**0.314**	^{abcdj}0.569	^{d}0.669	0.638	0.789	^{c}0.230	0.384
f	***TK-Sparse*** $\lambda = 1 * 10^{-3}$	19 %	^{abcdhj}**0.373**	^{abcdhj}**0.314**	^{abcdhj}**0.570**	^{cd}0.691	^{bcdeg}**0.658**	^{dceg}**0.840**	^{c}0.232	^{bcdegh}**0.400**
g	***TK-Sparse*** $\lambda = 3 * 10^{-3}$	12 %	^{abcdj}0.372	^{abcdj}**0.314**	^{abcdj}0.568	^{d}0.665	0.632	0.758	0.220	^{d}0.382
h	***TK-Sparse*** $\lambda = 6 * 10^{-3}$	26 %	^{abcdj}0.371	^{bcdj}0.312	^{bcdj}0.567	0.681	0.653	0.821	0.231	0.391
i	***TK-Sparse*** $\lambda = 9 * 10^{-3}$	18 %	^{abcdhj}0.371	^{abcdj}0.312	^{bcdj}0.567	^{bcdeg}**0.714**	^{bcdeg}0.657	^{dcg}0.827	0.227	^{bdegh}**0.400**
j	***TK-Sparse*** $\lambda = 2 * 10^{-2}$	43 %	0.350	0.293	0.542	^{bcdgh}0.705	^{bcdeg}0.657	^{dcg}0.832	^{c}0.239	^{d}0.395

3 EXPERIMENT DESIGN

We conduct our experiments using the MS MARCO passage collection [2] with 8,84 million passages and two distinct query test sets: The limited-judged DEV set and the densely-judged TREC-2019 from the Deep Learning track [3][1]. The DEV set contains 48,598 queries, with 1.06 judgments per query on average, whereas the TREC set contains 43 queries with 215 judgments per query on average. We used a binarization point of 2 from the graded TREC scale and the provided initial ranking list. We conduct statistical significance tests with a Wilcoxon signed-rank test with $p < 0.05$.

We capped the passages at 200 and the queries at 30 tokens. We use a batch size of 32 and the Adam optimizer with a learning rate of 10^{-4} for representation and sparsity; 10^{-3} for kernel-pooling. Our early stopping is based on the nDCG@10 value of our validation set. Each neural model re-ranks the top 1000 BM25 results.

For the parameter settings of the evaluated models we followed Hofstätter et al. [9]. For TK & TK-Sparse we used a 2-layered Transformer with 300 dimensional GloVe embeddings. For kernel-pooling we use the default number of 11 kernels from −1 to +1 and standard deviation of 0.1 for all kernels. For the baselines BM25 (computed by Anserini [20]), DUET [15], CO-PACRR [11], and CONV-KNRM [5] we utilize their respective default parameters. We chose to restrict us to these baselines, as they have been shown to be the most effective neural ranking models [8], that allow to offline pre-compute document representations, to speed up query response time. Because our work is concerned with improving the efficiency in this scenario. This excludes BERT-based re-ranking, as it always needs to compute document representations for every query [16].

For the traditional stopword list we utilized the Lucene stopwords with 33 common words[2], additionally we selected the top 25 and 50 occurring tokens from the MS MARCO corpus to evaluate domain specific stopwords.

[1]We note that BM25, DUET, TK and BERT models participated in the TREC-2019 evaluation, therefore they have an advantage of not being affected by pool bias [12].
[2]Taken from the EnglishAnalyzer class

4 RESULTS

We present our results in Table 1. The first section contains non-TK baselines, the second section TK-based baselines with traditional stopwords removed and the third section contains results of TK-Sparse with our novel stopword component.

Training the TK model on documents preprocessed with a traditional stopword filtering component performs worse than the TK model without filtering on all evaluation metrics on both query sets. The more words get filtered the worse the results become. On the other hand it also increases the model's efficiency, as less tokens need to be processed.

When we look at the sparsity of different TK-Sparse configurations, we can see that the word removal between 12-26% does not follow the ordering of λ; only lower and higher λ values have a more forceful impact on the sparsity. In Figure 1 we see, that even though the different configurations converge to different sparsity levels, many individual points overlap. We select the best model instance with early stopping, based on the nDCG@10 value alone. In future work we plan to also incorporate a sparsity target, to be able to control the desired sparsity value.

From the results of TK-Sparse $\lambda = 8 * 10^{-4}$, with very little stopword removal, we deduce that the limited DEV set profits from the added salience value of r_j per retained term, although the high-quality TREC queries do not profit. This might constitute a small overfitting, which is lifted as soon as more words are removed. We plan to further investigate the differences between limited and dense judgments in future work.

In general, the sparsity range around 20% offers the best results overall, for both limited and dense judgments. For the TK-Sparse $\lambda = 2 * 10^{-2}$ configuration, we observe an interesting pattern with our judgments: The limited DEV results fall down to the same level as the worst traditional stopword list with a similar stopword ratio. However, in stark contrast to the low traditional *CollectionTop50* result on the TREC judgments, the contextualized stopword approach with 43 % of the terms removed shows very strong results on the

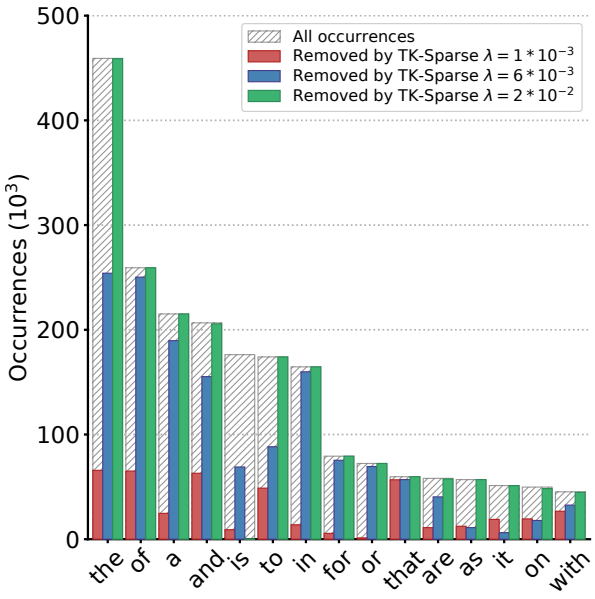

Figure 2: Occurrences and contextualized removal count of the 15 most frequent Lucene stopwords on the validation set.

high-qualitative judgments. We assume this is due to the higher robustness of multiple relevant judged documents, that penalizing a single document with wrongly removed terms has not a significant impact on the overall result.

5 CONTEXTUAL STOPWORD ANALYSIS

The dependence of our stopwords on their context requires us to analyze their distribution to get an insight into the inner workings of our stopword component. In Figure 2 we show the occurrences of the 15 most frequent words from the Lucene stopword list and the number of removals by different TK-Sparse configurations. For the two TK-Sparse models, which in total remove less or the same amount of words as Lucene's list ($\lambda = 1 * 10^{-3}$ and $6 * 10^{-3}$), we can clearly observe a much less aggressive removal of stopwords. Only the $\lambda = 2 * 10^{-2}$ configuration removes most of Lucene's stopwords, as it removes many more words in total. Notable is the seldomly removed term "is" (which is consistent across TK-Sparse models).

To showcase the difference between two sparsity-levels we highlight their selected stopwords in a passage in Figure 3. We can see that $\lambda = 2 * 10^{-2}$ model is more aggressive in removing words. Notable are directional instructions ("to be", "dna to mrna" and "mrna in the nucleus") that are retained in the more conservative TK-Sparse instance.

To conclude, we summarize that contextual and traditional stopwords have a wide area of overlap in the terms they target. However contextual stopwords have a broader reach and also remove fewer occurrences of the traditional stopwords.

6 CONCLUSION

In this paper we presented a new approach to the use of stopwords in IR: Learning to remove words from a document with a sparse vector, depending on the local context of the word in a sequence.

TK-Sparse $\lambda = 6 * 10^{-3}$

lesson summary . 1 the first step in protein production is the transcription of dna to mrna in the nucleus . 2 the next organelles involved are ribosomes . 3 the golgi apparatus is involved for proteins destined to be transported to the cell membrane and acts as the cell 's post office .

TK-Sparse $\lambda = 2 * 10^{-2}$

lesson summary . 1 the first step in protein production is the transcription of dna to mrna in the nucleus . 2 the next organelles involved are ribosomes . 3 the golgi apparatus is involved for proteins destined to be transported to the cell membrane and acts as the cell 's post office .

Figure 3: Example of contextualized stopwords in red & underlined for two sparsity models (passage id: 5035398).

With TK-Sparse we integrated our stopword component into the TK neural re-ranking model. Our results demonstrate that traditional stopwords do not have a positive impact on the effectiveness of neural re-ranking models. TK-Sparse shows significantly better effectiveness results with up to 40% of the words removed for more space-efficient document pre-computation.

REFERENCES

[1] Leif Azzopardi and Vishwa Vinay. 2008. Retrievability: an evaluation measure for higher order information access tasks. In *Proc. of CIKM*.
[2] Payal Bajaj, Daniel Campos, Nick Craswell, et al. 2016. MS MARCO : A Human Generated MAchine Reading COmprehension Dataset. In *Proc. of NIPS*.
[3] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2019. Overview of the TREC 2019 deep learning track. In *TREC*.
[4] Zhuyun Dai and Jamie Callan. 2019. Context-aware sentence/passage term importance estimation for first stage retrieval. *arXiv preprint 1910.10687* (2019).
[5] Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. 2018. Convolutional Neural Networks for Soft-Matching N-Grams in Ad-hoc Search. In *Proc. of WSDM*.
[6] Ljiljana Dolamic and Jacques Savoy. 2010. When stopword lists make the difference. *Journal of the American Society for IST* 61 (2010).
[7] Jiafeng Guo, Yixing Fan, Qingyao Ai, and Bruce Croft. 2016. A Deep Relevance Matching Model for Ad-hoc Retrieval. In *Proc. of CIKM*.
[8] Sebastian Hofstätter and Allan Hanbury. 2019. Let's measure run time! Extending the IR replicability infrastructure to include performance aspects. In *Proc. of OSIRRC*.
[9] Sebastian Hofstätter, Markus Zlabinger, and Allan Hanbury. 2020. Interpretable & Time-Budget-Constrained Contextualization for Re-Ranking. In *Proc. of ECAI*.
[10] Sebastian Hofstätter, Markus Zlabinger, and Allan Hanbury. 2020. Neural-IR-Explorer: A Content-Focused Tool to Explore Neural Re-Ranking Results. In *Proc. of ECIR*.
[11] Kai Hui, Andrew Yates, Klaus Berberich, and Gerard De Melo. 2018. Co-PACRR: A Context-aware Neural IR Model for Ad-hoc Retrieval. In *Proc. of WSDM*.
[12] Aldo Lipani. 2016. Fairness in information retrieval. In *Proc. of SIGIR*.
[13] Aldo Lipani, Mihai Lupu, Akiko Aizawa, and Allan Hanbury. 2015. An Initial Analytical Exploration of Retrievability. In *Proc. of ICTIR*.
[14] Tomáš Mikolov. 2012. Statistical language models based on neural networks. *Ph.D. thesis, Brno University of Technology* (2012).
[15] Bhaskar Mitra and Nick Craswell. 2019. An Updated Duet Model for Passage Re-ranking. *arXiv preprint arXiv:1903.07666* (2019).
[16] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *arXiv preprint arXiv:1901.04085* (2019).
[17] Stephen Robertson. 2004. Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of documentation* (2004).
[18] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-End Neural Ad-hoc Ranking with Kernel Pooling. In *Proc. of SIGIR*.
[19] Liu Yang, Qingyao Ai, Jiafeng Guo, and Bruce Croft. 2016. aNMM : Ranking Short Answer Texts with Attention-Based Neural Matching. In *Proc. of CIKM*.
[20] Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the use of Lucene for information retrieval research. In *Proc. of SIGIR*.
[21] ANK Zaman, Pascal Matsakis, and Charles Brown. 2011. Evaluation of stop word lists in text retrieval using Latent Semantic Indexing. In *Proc. of DIM*.
[22] Hamed Zamani et al. 2018. From Neural Re-Ranking to Neural Ranking: Learning a Sparse Representation for Inverted Indexing. In *Proc. of CIKM*.