CHOPPY: Cut Transformer for Ranked List Truncation

Dara Bahri*
Google Research
dbahri@google.com

Yi Tay
Google Research
yitay@google.com

Che Zheng
Google Research
chezheng@google.com

Donald Metzler
Google Research
metzler@google.com

Andrew Tomkins
Google Research
tomkins@google.com

ABSTRACT

Work in information retrieval has traditionally focused on ranking and relevance: given a query, return some number of results ordered by relevance to the user. However, the problem of determining how *many* results to return, i.e. how to optimally truncate the ranked result list, has received less attention despite being of critical importance in a range of applications. Such truncation is a balancing act between the overall relevance, or usefulness of the results, with the user cost of processing more results. In this work, we propose CHOPPY, an assumption-free model based on the widely successful Transformer architecture, to the ranked list truncation problem. Needing nothing more than the relevance scores of the results, the model uses a powerful multi-head attention mechanism to directly optimize any user-defined IR metric. We show CHOPPY improves upon recent state-of-the-art methods.

CCS CONCEPTS

• **Information systems** → **Presentation of retrieval results**; **Retrieval effectiveness**.

KEYWORDS

ranked list truncation; neural networks; Transformer; information retrieval; deep learning

ACM Reference Format:
Dara Bahri, Yi Tay, Che Zheng, Donald Metzler, and Andrew Tomkins. 2020. CHOPPY: Cut Transformer for Ranked List Truncation. In *43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3397271.3401188

1 INTRODUCTION

While much of the work in information retrieval has been centered around ranking, there is growing interest in methods for *ranked list truncation* - the problem of determining the appropriate cutoff k of candidate results [1, 9]. This problem has garnered attention in fields like legal search [14] and sponsored search [3, 16], where there could be a monetary cost for users looking into an irrelevant tail of documents or where showing too many irrelevant ads could result in ad blindness. The fundamental importance of this problem has led to development of methods that are automatically able to learn k in a data-driven fashion [9]. The focus of this paper is to design more effective models for accurate and dynamic truncation of ranked lists.

The present state-of-the-art for this task is BiCut [9], a recurrent-based neural model that formulates the problem as a sequential decision process over the list. BiCut trains a bidirectional LSTM [8] model with a predefined loss that serves as a proxy to the user-defined target evaluation metric. At every position in the ranked list, BiCut makes a binary decision conditioned on both forward and backward context: to continue to the next position, or to end the output list.

While BiCut outperforms non-neural methods [1], we argue it has several drawbacks. Firstly, the model is trained with teacher-forcing, i.e. with ground truth context, but it is deployed auto-regressively at test time, where it is conditioned on its own predictions. Thus, the model suffers from a train / test distribution mismatch, often referred to as exposure bias [12], resulting in poor model generalization. Secondly, the loss function used does not capture the mutual exclusivity among the candidate cut positions. In other words, the loss does not capture the condition that the list can only be cut in at most one position. Furthermore, the proposed training loss is unaligned with the user-defined evaluation metric. Last but not least, BiCut employs BiLSTMs which are not only slow and non-parallelizable, but also do not take into account global long-range dependencies.

This paper proposes CHOPPY, a new method that not only ameliorates the limitations of the BiCut model but also achieves state-of-the-art performance on the ranked list truncation task. Our method comprises two core technical contributions. The first is a Transformer model [15] that is able to capture long-range dyadic interactions between relevance scores. To the best of our knowledge, this is the first successful application of self-attentive models on scalar ranking scores. The second technical contribution is the development of a loss function that optimizes the *expected* metric value over all candidate cut positions. Overall, CHOPPY not only improves the predictive performance on this task but also improves the model inference speed by > 3 times.

Our contributions. The key contributions of the paper are summarized as follows:

- We frame the ranked list truncation task as modeling the joint distribution among all candidate cut positions, and we

*Corresponding author

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SIGIR '20, July 25–30, 2020, Virtual Event, China
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401188

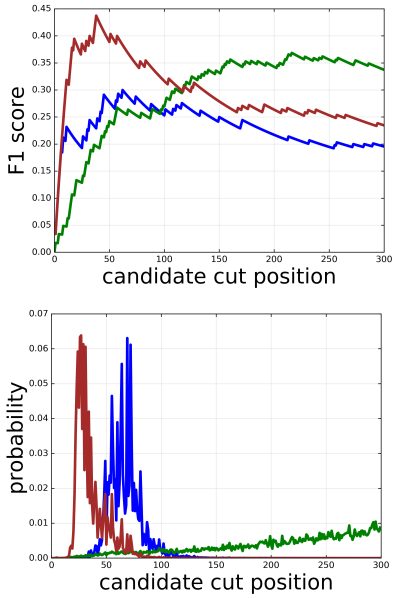

Figure 1: Top: F1 at various cut positions for 3 training queries from Robust04 BM25. Bottom: CHOPPY's softmax predictions for the same queries.

construct our training loss to be the *expected* metric value over cut positions, for *any* choice of user-defined metric, such as F1, precision, or discounted cumulative gain. As such, our training loss and evaluation metric are fully aligned.

- We propose CHOPPY, a Cut Transformer model that achieves a state-of-the-art 11.5% relative improvement over the BiCut model. To predict the joint distribution over candidate cut positions, our method learns a positional embedding and leverages expressive bidirectional multi-headed attention.

2 RELATED WORK

Across the rich history of information retrieval research, there has been extensive work focused on modeling score distributions of IR systems. Early work in this area primarily focused on fitting parametric probability distributions to score distributions [1, 10]. This is often achieved by making the assumption that the overall distribution can be expressed as a mixture of a relevant and a non-relevant distribution. The expectation-maximization (EM) algorithm is often adopted to learn the parameters.

There has been considerable recent interest in adopting machine learning based models to optimize and improve the ranked list truncation problem. For instance, cascade-style IR systems [17] seek to achieve a balance between efficiency and effectiveness. Notably, [5] investigates a number of machine learning approaches for learning dynamic cutoffs within cascade-style ranking systems. Another recent study investigated how to leverage bidirectional Long Short-Term Memory (LSTM) models to identify the best position to truncate a given list [9]. This model, BiCut, can be considered the present state-of-the-art approach.

Our work is closely related to the task of query performance prediction [4]. In this task, the objective is to automatically determine the effectiveness of a given query. This could be leveraged to determine the optimal set of results to the user for any given measure. Methods for query performance prediction include pre-retrieval-based approaches [7], relevance-based approaches [4, 19], and neural approaches [18].

A system that determines the best number of results to display to users has the potential to benefit a wide number of applications. For example, in sponsored search, displaying too many irrelevant ads to users may cause frustration, resulting in so-called *query blindness*. This motivated research that investigated whether any ads should be displayed at all [3]. It is also easy to see that a similar and related problem formulation is to determine how many ads should be displayed to the users [16]. Moreover, determining the optimal number of ranked results is also important in a number of other IR applications such as legal e-discovery [14], where there is an significant financial or labor cost associated with reviewing results. Finally, the ability to calibrate scores across queries and different corpora has also been studied in the context of federated search tasks [13] such as meta-search [11].

3 CHOPPY

We now describe CHOPPY, our proposed Cut Transformer approach for ranked list truncation.

Let $(r_1, \ldots, r_n)$ denote the sequence of results, ranked in decreasing order of relevance, and let r_i have relevance score s_i and ground truth relevance label y_i (1 if relevant, -1 if non-relevant). We now describe our model piecemeal.

3.1 Transformer Layer

We briefly review the Transformer layer. In our work, we let all model dimensions be d. Let $X \in \mathbb{R}^{n \times 3}$ represent the input to the layer and $W_@, W_\cdot, W_E \in \mathbb{R}^{3 \times 3}$. We define

$$\text{Attn}\ (X; W_@, W_\cdot, W_E) = \text{Act}\ (XW_@\, W_\cdot^T\, X^T)\ (XW_E)$$

where Act is an activation function. As done in [15], we take it to perform row-wise softmax and normalize by $\sqrt{d}$. Attention is often augmented using multiple heads. In multi-headed attention, the model dimension is split into multiple heads, each one performing attention independently, and the result is concatenated together. Let h be the number of heads and suppose d is divisible by h. Then,

$$\text{MultiAttn}\ (X; W_@, W_\cdot, W_E) = \underset{1 \le i \le h}{\text{rConcat}}\ \text{Attn}\ (X; W_@^{(i)}, W_\cdot^{(i)}, W_E^{(i)})$$

where $W_@^{(i)}, W_\cdot^{(i)}, W_E^{(i)} \in \mathbb{R}^{3 \times (3/h)}$. rConcat performs row-wise concatenation. The output of multi-headed attention is

$$A = \text{LayerNorm}\ (X + \text{MultiAttn}\ (X; W_@, W_\cdot, W_E))$$

where LayerNorm is layer normalization [2]. Finally, the output of the Transformer layer is

$$O_{\text{trans}} = \text{LayerNorm}\ (A + \text{rFF}\ (A))$$

where rFF applies a single learnable feed-forward layer with ReLU activation to each row of A.

3.2 Positional Embedding

The vanilla Transformer incorporates positional encoding by adding fixed sinusoidal values to the input token embeddings. As the token embeddings are trained, they have the flexibility to learn how to

best utilize the fixed positional information. In our setting however, the inputs are fixed 1-dimensional relevance scores. Attempting to apply a Transformer layer directly on the raw scores can limit its complexity. To that end, we introduce a learnable positional embedding $P \in \mathsf{R}^{=\times(3-1)}$ and feed in $X = \text{rConcat}[\mathsf{s}, P]$ to the the first Transformer layer only, where s is the column vector of relevance scores.

3.3 Loss

So far, Choppy takes the n-length vector of scores, augments it with a positional embedding, and feeds the result into n_{layers} Transformer layers. This produces $O_{\text{trans}} \in \mathsf{R}^{=\times 3}$. We arrive at the output of Choppy by applying a final linear projection followed by a softmax over positions:

$$\mathsf{o} = \text{Softmax}\left(O_{\text{trans}} W_{>}\right)$$

where $W_{>} \in \mathsf{R}^{3\times 1}$. We interpret the output o to be a probability distribution over candidate cutoff positions. More concretely, we take $\mathsf{o}_8 = \text{Prob}\left[(\mathsf{r}_1, \ldots, \mathsf{r}_8)\right]$. Let C be any user-defined evaluation metric that should be maximized, such as F1 or precision. For each training example j and every candidate cutoff position i we compute $C_8(\mathsf{y}^{(9)})$, the value of the metric if the result list were to be truncated at position i, using the ground-truth relevance labels. Our proposed loss follows as:

$$L\left(\mathsf{s}^{(9)}, \mathsf{y}^{(9)}\right) = -\sum_{8=1}^{\tilde{\Theta}} \mathsf{o}_8\left(\mathsf{s}^{(9)}\right) C_8\left(\mathsf{y}^{(9)}\right)$$

$$= -\mathsf{E}_{/\sim\text{Categorical}\left(\mathsf{o}\left(\mathsf{s}^{(j)}\right)\right)} C_/\left(\mathsf{y}^{(9)}\right).$$

With this loss, our model learns the conditional joint distribution over candidate cut positions that maximizes the expected evaluation metric on the training samples. We depict the loss and the predicted distribution for a few training samples in Figure 1. We see that the model tends to weight positions according to their corresponding metric value. At test time we choose to cut at the argmax position. Note that unlike BiCut, our loss has no tune-able hyperparameters.

4 EXPERIMENTS

This section describes our experimental setup and results.

4.1 Dataset

We evaluate our method using the TREC collection Robust04, used in the TREC 2004 Robust Track. It consists of 250 queries over 528k news articles, where each query has 1000 total results and an average of 70 relevant ones. This is the same dataset used in [9]. We use a random 80/20 train/test that achieves comparable performance to the reported results in [9]. We evaluate the efficacy of our truncation model using two different retrieval approaches - BM25, a traditional tf-idf based model, and DRMM [6], a neural model.

4.2 Baselines

We evaluate our method against the following baselines:

- **Fixed-k** returns the top-k results for a single value of k across test queries.

	BM25		DRMM	
	F1	DCG	F1	DCG
Oracle	0.367	1.176	0.375	1.292
Fixed-$:$ (5)	0.158	-0.261	0.151	0.010
Fixed-$:$ (10)	0.209	-0.708	0.197	-0.407
Fixed-$:$ (50)	0.239	-5.807	0.261	-5.153
Greedy-$:$	0.248	-0.116	0.263	0.266
BiCut	0.244	-	0.262	-
Choppy	**0.272**	**-0.041**	**0.268**	**0.295**
Rel. % Gain	+11.5%	-	+2.29%	-

Table 1: Average F1 and DCG performance on Robust04. Choppy achieves state-of-the-art performance. "Gain" reports relative performance gain over BiCut model.

- **Greedy-k** chooses the single k that maximizes C over the training set.
- **Oracle** uses knowledge of each test query's true label to optimize k. It represents an upper-bound on the metric performance that can be achieved.
- **BiCut** [9] learns a multi-layer bidirectional LSTM model on the entire training set, taking the score sequence as inputs. At position i of the result list, the model predicts probability p_8 to continue and probability $1 - p_8$ to **end**. At inference time, the cutoff is made before the first occurrence of **end**.

4.3 Setting

We report F1 and Discounted Cumulative Gain (DCG) scores where we define DCG to penalize negative, or non-relevant results:

$$\text{DCG}_= = \sum_{8=1}^{\tilde{\Theta}} \frac{y_8}{\log_2(i+1)},$$

We need to deviate from the usual definition of DCG since the usual definition always increases monotonically with the length of the returned ranked list and so the optimal solution under this definition would be to not truncate at all. For methods that optimize F1 or DCG, we report the performance of the model when it is optimized specifically for that metric. Note that DCG is unsupported by BiCut.

For Choppy, we blithely set $n_{\text{layers}} = 3$, h (# heads) $= 8$, and $d = 128$ across all settings, without any tuning. We optimize the aforementioned custom loss function using Adam with default learning rate 0.001, and a batch size of 64. As in [9], we only consider the top-300 candidate results of each query.

5 RESULTS AND DISCUSSION

5.1 Results

As shown in Table 1, Choppy achieves a significant improvement over BiCut for both metrics and both retrieval types. This improvement arises from Choppy's ability to model the joint distribution over *all* candidate cut positions and its direct optimization of the evaluation metric. Furthermore, the attention mechanism is able to effectively capture correlations between scores far apart in ranked order. This is in contrast to LSTMs, as used in BiCut, whose degradation with larger sequence length is well known.

5.2 Ablation Study

Choppy has three hyperparameters: the model dimension d, the number of heads h, and the number of Transformer layers n_{layers}.